Table Header Detection

Extracting Header Rows From Web Tables using Machine Learning

Huebscher, Leonardo Waack, Juliane

Hasso Plattner Institute
Potsdam, Germany

ABSTRACT

There is a vast amount of structured information in the form of HTML tables available on the web. To make this information processable for answering queries, we need to extract schema information about the table.
We test different machine learning models, features and feature encodings of web tables to determine how to differentiate between header, data, and other rows in relational tables crawled from Wikipedia. In our testing, we were able to reproduce the good results Adelfio and Samet got in their paper Schema Extraction for Tabular Data on the Web by using Conditional Random Fields and logarithmic binning [1]. We also showed, that on our data the choice of machine learning algorithm and the extra step of using logarithmic binning did not have a large effect on the performance. Tests using a continuous encoding and Random Forests returned similar results.

CCS CONCEPTS

• **Information systems → Data extraction and integration**;

KEYWORDS

information retrieval, web tables, table header extraction

1 INTRODUCTION

The internet is full of heterogenous information. While most of it is unstructured, there also is a large amount of structured data. In 2008, Cafarella et al. extracted about 158 million HTML tables containing relational data from a Google web crawl [3].
Contrary to tables in databases, these tables are missing schema information, though, which is needed to be able to query and analyze them effectively. As Cafarella et al. put it: "Even the largest corpus is useless if we cannot query it."
To be able to make use of the vast amount of data contained in HTML tables all over the internet, we need to extract their schema information automatically.

What a table's columns contain is necessary information to enable effective querying on that table. This information is at least partially contained in the headers of these columns. While about 70% of web tables have a "simple" structure and are comprised of exactly one header row followed by only data rows, this still leaves a large amount of tables whose more complex structure needs to be identified [1]. In HTML tables it is often unclear which cells or rows contain headers and which contain data.
This can lead to problems because without this information content of cells, for example numbers, can become meaningless. Additionally, treating header cells as data cells will cause errors in aggregates. Most web tables are written to be read and understood by human readers, not machines. As a result of this, headers can be marked by many different features like background color, font-size and boldness, but also contextual information. If the first element in a column contained the string "Year", for example, and all others were four digit numbers, most humans would not need any styling information to determine that the first cell is most likely the header while the others contained data. What features are used to distinguish headers can vary widely and conventions like the use of th-tags to mark header rows are not enforced and therefore often disregarded [7].
The goal of this paper is to find an automatic way to distinguish between header and data rows in an HTML table. Our algorithm takes an HTML table as input and creates a sequence of labels as the output. Each row gets labeled as either a header row, a data row or neither of the two. The latter is called "other row" and is used for rows containing notes about the table as well as rows containing aggregates and other content that does not correspond to the above headers.

2 RELATED WORK

Several different approaches to detecting headers in web tables have already been proposed and tested:
Chen et al. use a rule-based algorithm to determine whether a table is oriented row-wise or column-wise and subsequently find the corresponding headers for every cell in the table [4].
Cafarella et al. use machine learning to find headers in tables [3]. They do however discard tables with more than one header row as non-relational, so their detector just classifies whether the first row in a table is a header or not.
Pinto et al. use Conditional Random Fields [5] to assign table rows into different categories, among them DATAROW and TABLEHEADER [8]. Adelfio and Samet use a similar approach on web tables, but with altered features and the addition of logarithmic binning to combine cell features into row features.
This work is primarily based on their paper Schema extraction for tabular data on the web [1].

We used the corpora of Lehmberg et al. [6] and the Web Tables Corpora (WTC)[1] as a basis for crawling our own data and testing our models.

3 APPROACH

This section describes the different approaches to automatic header detection tested in this paper. First, the solution based on the Schema Extraction paper by Adelfio and Samet and its differences to the paper are explained in section 3.1 [1]. The second part is about a different machine learning model as well as the adaptations

[1] http://webdatacommons.org/webtables/index.html

- **isTHorInTHead**
- **hasColSpan**
- **hasRowSpan**
- **isPartiallyBold**
- **isCompletelyBold**
- isItalic
- isUnderlined
- isCenterAligned
- isTotal
- isShortText
- isLongText
- isText
- isDate
- isNumeric
- isEmpty
- isMerged

Figure 1: The list of used boolean features without the similarity features. Features that differ from the once described in [1] are marked in bold text

made to the features and their encoding in contrast to the previous method.

3.1 Conditional Random Fields

After normalizing the tables so that every row contains the same amount of cells, the boolean features described by Adelfio and Samet in their paper are extracted for each cell [1] . During the normalization process every cell spanning more than one column or row is duplicated to ensure all rows and columns contain the same number of cells. This is necessary to calculate the similarity features for neighbouring cells, as described by Adelfio and Samet [1].
The feature "isTHOrInTHead" is additionally calculated, to compensate for the sometimes missing styling information from css-files. The thresholds set for the two features "isLongText" and "isShortText" are 20 and 255.

The cell features are combined to row features using the logarithmic binning approach specified in [1]. This encoding is used to abstract from the table size while retaining a higher level of similarity between tables of similar size. The idea is based on the intuition that to classify a row with many cells it is more helpful to compare it to other rows with similarly many cells as opposed to a row with only 2 cells, for example. This difference in number of cells in a row is not considered when using continuous encoding like we did with the random forests approach described in 3.2. Here two rows in which half the cells have a certain feature are considered similar independently of the number of cells they contain.
To clearly distinguish between these kinds of rows, Adelfio and Samet use a direct encoding in which the rows are put into bins based on their number of cells (r) and the number of cells for which the currently examined feature is true (c). The bins are represented by two indices, which are calculated logarithmically based on c and r. The higher these numbers are, the larger the difference between two rows that belong to the same bin can be.
A weakness of the original features is that partial boldness of a cell content can not be accurately represented in the feature space. Furthermore the "isMerged" feature does not distinguish between column- and rowspans, which have different semantics. Thus we introduce the new features "isPartiallyBold", "isCompletelyBold", "hasColSpan" and "hasRowSpan". The results of this approach can be seen in section 4.3

The similarity features for each of the mentioned features in Figure 1 are calculated as described by Adelfio and Samet [1]:
For every cell c and every feature f with $f(c)$ being the value of the feature for the cell c, similarity features comparing c with its two neighbouring cells in the same column are calculated.
Let c' be one of the neighbouring cells to c. f_a and f_b are the formulas to calculate the similarity features for c and c'.

$$f_a(c) = f(c) \wedge f(c')$$
$$f_b(c) = f(c) \vee \neg f(c')$$

3.2 Random Forests

Since the usage of logarithmic binning combined with conditional random fields was the key contribution of Schema Extraction Paper by Adelfio and Samet [1] that we based our approach on, we were interested to see how well the features worked with different machine learning algorithms.
To test this we used the improved features as an input for random forests and normalized the features for each row instead of binning them. This was done by counting how many cells in a row have a certain feature and dividing this number by the number of cells in the row. We decided to use random forests, because they are precise but not prone to overfitting [2].
The random forests model only considers one row's features when deciding on a label for that row. This is in contrast to the conditional random fields, which predict a sequence of labels for a whole table and therefore indirectly take into account the position of a row in the table. To compensate for this lack of positional awareness of the random forests, the feature "normalizedRowIndex" is added and used in the tests. The scikit-learn library does not accept any input containing non-numeric values like NaN (not a number) for random forests. These numbers occur in the similarity features for the first and last row of a table because they do not have an upper or lower row to be compared to. To adjust the input to the library's needs, any occurrence of NaN is replaced with -1, a number no other feature could be assigned.

4 EVALUATION

This section is about evaluating the performance of different table header detection algorithms and comparing them. Firstly, the testing environment and data set that are being used for training and testing of the machine learning models will be introduced. Secondly, the typical metrics for models such as precision, recall, and f1-score are being evaluated. These metrics are used to compare the chosen baseline to the related approach that is based on the Schema Extraction paper[1] which got published by Adelfio and Samet. Furthermore, we will test a different machine learning classifier. The code of our implementations and evaluations can be found on GitHub[2]

4.1 Experimental Setup

Similar to the Schema Extraction paper[1], we use 10-fold cross-validation to calculate the predictive performance. We will compare the following scenarios:

[2]https://github.com/deeps96/web-tables-header-detection

- **Baseline**
  As stated in the WebTables Paper[3], about 70% of all relational web tables are considered *simple*. Simple tables only consist of one header row followed by several data rows. As the Schema Extraction Paper[1] pointed out, some approaches "[...] rely on the high quantity of data tables that have simple structure, in order to avoid dealing with those data tables with more complex structure"[1] We think that this should be a good baseline to compare to.
- **AdelSam**
  This approach reflects the original implementation that is described in the Schema Extraction paper[1]. We want to examine whether the stated predictive performance metrics that have been mentioned in the paper, hold for our data set.
- **Improved AdelSam**
  As the name already indicates, this is a slightly modified implementation of the AdelSam algorithm that aims at further improvements.
- **Simplified**
  Instead of using the more complex logarithmic binning together with Conditional Random Fields, which both can be considered as a key contribution of the Schema Extraction paper[1], we feed the same set of extracted cell features into an approach that is based on random forests.

4.2 Data Set

We faced several challenges while searching for a proper data set. We took a look into *The Dresden Web Table Corpus*[3], *Web Table Corpora*[4] and *WikiTables*[5]. The main problem with all data sets is the lack of styling information. In some of the data sets, f.e. the WikiTables, some style attributes have been removed due to normalization. The existence of style information is crucial to make the feature extraction described in section 3.1 work. Therefore we decided to crawl our own set of web tables. As a starting point, we use the Wikitables data set that got created as part of the paper TabEL: Entity Linking in Web Tables in 2015. It contains roughly 1.6 million relation web tables and includes the HTML for each of them. However, important style attributes have been cropped out. For our crawling process, we fetched the pages from the data set, since it is very likely that the pages contain at least one relational web table. As some of the CSS stylings comes from CSS classes that have been defined externally, the algorithm needs to in-line the style attributes into the HTML tags. In the last step of the data set creation process, we extracted the tables including all style information. We crawled 5,325 pages from Wikipedia, resulting in 12,258 extracted web tables.

Since the algorithms are using supervised machine learning for row type classification, we needed to label the crawled data set. The initial goal was to label 1,000 tables. Most of the relational web tables only have one header row, followed by multiple data rows. Therefore we made a more thoughtful selection of tables before we start labeling them. We decided to have four equally sized shares of 250 tables for each of the following category:

- 250 tables with no *<th>* and no *<thead>* HTML tags
- 250 tables containing at least one merged cell, indicated by the *colspan* or *rowspan* attribute
- 250 tables where at least one cell contains bold text and is not placed inside the header
- 250 tables chosen randomly

All categories are distinct from each other and are taken from 741 different Wikipedia pages. During the labeling process, we had to discard 71 tables. This is because our crawling process did not discard non-relational tables such as matrices or other table types. On average the resulting tables have a size of 15 rows. In total the data set contains 14,192 rows, where 12,277 (86.5%) rows are labeled as *data* row, 1,091 (7.7%) as *header* row and 824 (5.8%) as type *other*. The row types are not equally distributed per table. Additionally, on every single test-set, the average can vary. Table 1 shows the minimum and maximum shares for each row type compared to the average across all ten test sets.

	min	max	average across data set
header	4.94%	11.63%	5.81%
data	76.83%	93.18%	86.51%
other	1.89%	11.54%	7.69%

Table 1: Distribution of row types across the data set

What immediately catches the reader's eye is, that the data rows are dominating the tables. As a consequence, the row types "other" and "header" are likely mislabeled as "data". We would expect that other, as well as header, are mainly confused with data.

4.3 Evaluation of the Table Header Detection

In the first scenario, we are evaluating the typical metrics of a machine learning-based classification problem for the AdelSam approach and compare them to the numbers that have been stated in the Schema Extraction paper[1]. To compute the predictive performance, we used ten-fold cross-validation for all of our measurements. The results are shown in table 2. The increment indicates the difference to the numbers stated by Adelfio and Samet if applicable.

With the above results, we can confirm, that the numbers they have stated in their paper, are reproducible on our Wikitables data set. While being already in a very good performance range for classification algorithms, it might be hard to further improve the precision and recall without over-fitting.

However, in the next step we evaluate if we were able to further improve the AdelSam algorithm with the described changes that we have made for the Improved AdelSam approach.

	precision	recall	f1-score	support
header	0.94 (+0.03)	0.93 (±0)	0.94	1117
data	0.98 (−0.01)	0.93 (±0)	0.99	12620
other	0.89	0.70	0.78	824

Table 2: Predictive performance of the AdelSam approach compared to the numbers stated in [1]

[3] https://wwwdb.inf.tu-dresden.de/misc/dwtc/
[4] http://webdatacommons.org/webtables/index.html
[5] http://websail-fe.cs.northwestern.edu/TabEL/

	precision	recall	f1-score	support
header	0.94	0.93	0.94	1117
data	0.98	0.99	0.99	12620
other	0.89	0.71	0.79	824

Table 3: Predictive performance of the Improved AdelSam approach

Table 3 shows a small performance shift from the data towards the header precision, which is in our interest. Still, the change is very small and probably statistically irrelevant. As the results of the AdelSam approach have been very good already, it is hard to prove the relevance of an improvement.

To get a better understanding of what the Improved AdelSam algorithm is struggling with, we generated a confusion matrix that is shown in table 4. We aggregated the labels based on the ten-fold cross-validation.

predicted / **labeled**	**header**	**data**	**other**
header	1042 (93.29%)	65 (5.82%)	10 (0.90%)
data	40 (0.32%)	12519 (99.20%)	61 (0.48%)
other	29 (3.52%)	212 (25.73%)	583 (70.75%)

Table 4: Confusion matrix of the Improved AdelSam approach

The table reveals, that the row types "header" and "other" mostly get confused with the row type "data". However, the biggest challenge seems to be the separation of "data" and "other". One reason for the high amount of wrong classifications could be that in some cases only semantic or contextual knowledge can help distinguish between these two. Such features are not part of the introduced algorithms and therefore not handled. This could be a good starting point for further investigations.

The AdelSam approach has been shown to work with a very good precision already, which makes it difficult to further improve on. While we focus on web tables only, the authors of the Schema Extraction paper[1] also included spreadsheets. As the introduced logarithmic binning and the use of conditional random fields seems to be complex for the classification, we want to find out, whether other ML approaches are also feasible.

	precision	recall	f1-score	support
header	0.96	0.90	0.93	1117
data	0.98	0.99	0.99	12620
other	0.80	0.74	0.77	824

Table 5: Predictive performance of the Simplified approach

Table 5 shows the results returned for the Simplified approach that is based on random forests. At first sight the numbers are quite similar to the AdelSam approach. Figure 2 show the f1-score of all presented methods to ease the comparison between them.

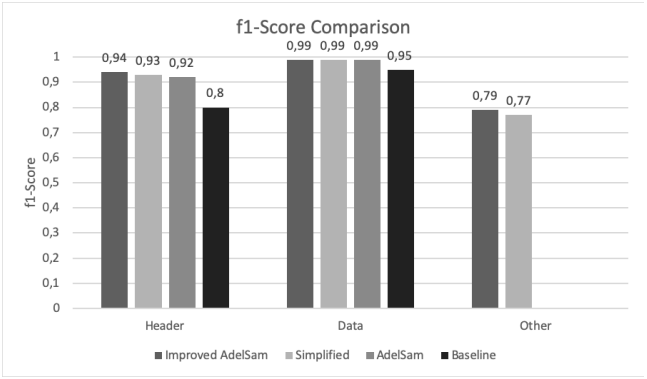

Figure 2: Comparison of the f1-score to the baseline

As we can see, the AdelSam and Improved AdelSam approaches can detect header rows more reliable compared to the simple baseline approach. It indicates that it could be indeed useful, to use a more complex approach when it comes to table header detection for web tables. The predominant data row type is limiting the expressiveness of the results, as they all perform similar to the baseline with regards to the data row type. The row type "other" is not comparable for AdelSam and the baseline, because both of them do not support this tag and are therefore not shown. Figure 2 also reveals that the results obtained with the Simplified approach are similar to Improved AdelSam if we merely focus on the f1-scores. When taking a closer look, we can see a shift of precision and recall for the row type other and header in the favor of the precision of the header row type. In the end, both approaches are comparable since the overall difference is small, considering a data set of roughly 1000 tables.

Additionally, the Simplified algorithm allows us to determine the importance of each feature. The far most important feature turned out to be the normalized row index. This could be due to the high number of simple tables and the dominance of header rows in the very beginning of a table. Other relevant features are mainly whether a cell is merged and whether it is center-aligned. Interestingly, the similarity features computed based on these features seem to have higher importance than the feature for just the cell itself.

4.4 Transfer of the Trained Model to Another Data Set

In the last section of the evaluation we want to transfer the machine learning model, that got trained on the Wikitables, to a table set from the Web Table Corpora (WTC). We used the crawler, described in section 4.2, to fetch the tables similarly. This ensured that both data sets have similar constraints, primary meaning that both data sets should have all style information. We crawled and labeled 202 tables while ensuring to have the same thoughtful selection of tables that we introduced for the Wikitables. The tables in this data set are generally smaller and only consist of eleven rows on average. Both data sets have a similar distribution of "header", "data", and "other" rows. Figure 3 shows the f1-scores of the Improved AdelSam approach and the Simplified approach. The approaches are trained

on the Wikitables data set and applied onto the WTC data set. To determine the f1-score, we use ten-fold cross-validation based on the Wikitables training data. Each fold is applied to all labeled tables from the WTC data set.

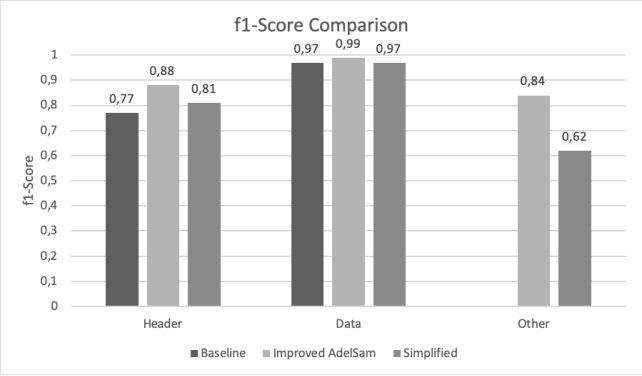

Figure 3: f1-score-comparison for the different approaches that have been trained on Wikitables and applied onto WTC

The above fig. 3 shows that the Improved AdelSam approach outperforms the Simplified one and the baseline. The numbers of the (unimproved) AdelSam approach are the same as for the Improved AdelSam. While "data" and "other" are somewhat comparable, the important "header" row type does not perform as expected. However, when labeling the WTC tables, we already noticed that the tables are somewhat repetitive. After counting the unique URLs the tables were coming from, we were surprised to see that we only had 106 distinct pages. The repetitiveness of the same table type might vitiate the meaningfulness of our results. Additionally, for some of the tables, it was even hard for a human to categorize a row. Sometimes we had to open the original HTML page and acquire some contextual information before making a decision. Considering the difficulties that we had, we are pleasantly surprised by how well the approach worked, especially compared to the simple baseline. The chart also indicates that a more complex approach for table header detection seems to be justified.

5 CONCLUSION

With our work we confirmed the good results Adelfio and Samet got in their paper and improved on them slightly. This was done by splitting the features "isBold" and "isMerged" to account for partial boldness and whether the colSpan or the rowSpan are merged. A large improvement would be very difficult because the results as measured by the f1-score are already very high for the identification of header and data rows.

The algorithm mostly struggled with differentiating between "data" and "other" rows. In some of the mislabeled cases, even a human needed contextual information to determine the correct row type. How to incorporate this information into the classification algorithm might be a good starting point for further research.

We were able to show, that the choice of machine learning algorithm and the logarithmic binning do not have a large effect on our data and similarly good results can be achieved by using random forests and a continuous encoding.

REFERENCES

[1] Marco Adelfio and Hanan Samet. 2013. Schema extraction for tabular data on the web. *Proceedings of the VLDB Endowment* 6 (04 2013), 421–432. https://doi.org/10.14778/2536336.2536343

[2] Jehad Ali, Rehanullah Khan, Nasir Ahmad, and Imran Maqsood. 2012. Random forests and decision trees. *International Journal of Computer Science Issues (IJCSI)* 9, 5 (2012), 272.

[3] Michael Cafarella, Alon Halevy, Daisy Wang, Eugene Wu, and Yang Zhang. 2008. WebTables: Exploring the power of tables on the web. *PVLDB* 1 (08 2008), 538–549. https://doi.org/10.14778/1453856.1453916

[4] Hsin-Hsi Chen, Shih-Chung Tsai, and Jin-He Tsai. 2000. Mining tables from large scale HTML texts. In *Proceedings of the 18th conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, 166–172.

[5] John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*. ICML, 282–289.

[6] Oliver Lehmberg, Dominique Ritze, Robert Meusel, and Christian Bizer. 2016. A large public corpus of web tables containing time and context metadata. In *Proceedings of the 25th International Conference Companion on World Wide Web*. International World Wide Web Conferences Steering Committee, 75–76.

[7] Rakesh Pimplikar and Sunita Sarawagi. 2012. Answering table queries on the web using column keywords. *Proceedings of the VLDB Endowment* 5, 10 (2012), 908–919.

[8] David Pinto, Andrew McCallum, Xing Wei, and W Bruce Croft. 2003. Table extraction using conditional random fields. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. ACM, 235–242.