Configurable Table Structure Recognition in Untagged PDF Documents

Alexey Shigarov
Matrosov Institute for System Dynamics and Control Theory of SB RAS
134 Lermontov st.
Irkutsk, Russia
shigarov@icc.ru

Andrey Mikhailov
Matrosov Institute for System Dynamics and Control Theory of SB RAS
134 Lermontov st.
Irkutsk, Russia
mikhailov@icc.ru

Andrey Altaev
Matrosov Institute for System Dynamics and Control Theory of SB RAS
134 Lermontov st.
Irkutsk, Russia
altaev@icc.ru

ABSTRACT

Today, PDF is one of the most popular document formats in the web. Many PDF documents are not images, but remain untagged. They have no tags for identifying the logical reading order, paragraphs, figures, and tables. One of the challenges with these documents is how to extract tables from them. The paper discusses a new system for table structure recognition in untagged PDF documents. It is formulated as a set of configurable parameters and ad-hoc heuristics for recovering table cells. We consider two different configurations for the system and demonstrate experimental results based on the existing competition dataset for both of them.

CCS Concepts

•**Applied computing** → **Document analysis;** *Document management and text processing;*

Keywords

table extraction, table structure recognition, untagged PDF documents, PDF document analysis, PDF accessibility

1. INTRODUCTION

Today, PDF is one of the most popular document formats in the web as can be measured by Google's "filetype:pdf" search. Many PDF documents are not images, but remain untagged. They have no tags for identifying the logical reading order, paragraphs, figures and tables. Nganji [4] estimates that 95.5% of scientific articles published by four leading publishers are untagged PDF documents. One of the important challenges with these documents is how to extract tables from them.

Table extraction typically consists of two main steps: *table detection*, i. e. recovering the bounding box of a table in a document, and *table structure recognition*, i. e. recovering its rows, columns, and cells. Many of existing methods for extracting tables from unstructured documents traditionally deal with only images or plain-text as source. They are considered in several surveys, including [1, 2]. These methods can be applied to PDF documents through converting PDF to these formats. However, this process leads to the loss of valuable information. Table extraction from PDF directly can provide better results. PDF is a richer representation of documents in comparison with images and plain-text. PDF documents can contain machine-readable text (text chunks with their positions, font characteristics, and order of appearance in a file), as well as vector graphics including table rulings. We expect that these features can allow to extract tables more accurately.

Several methods and tools for PDF table extraction are proposed in two last decades. Some of them are discussed in the surveys [1, 6, 7]. Ramel et al. [11] consider two techniques for detecting and recognizing tables from documents in an exchange format like PDF. The first is based on the analysis of ruling lines. The second is to analyze the arrangement of text components. Hassan et al. [5] expand these ideas for PDF table extraction. In the project TableSeer, Liu et al. [8] propose methods for detecting tables in PDF documents and extracting metadata (headers). They use text arrangement, fonts, whitespace, and keywords (e. g. "Table", "Figure"). Oro et al. [10] present PDF-TREX, a heuristic method where PDF table extraction is realized as building from content elements to tables in bottom-up way.

Yildiz et al. [15] propose a heuristic method for PDF table extraction using the 'pdftohtml'[1] tool for generating its input. Rastan et al. [12] consider a framework for the end-to-end table processing including the task of table structure recognition. They also use the 'pdftohtml' tool to prepare their input. However, this tool occasionally makes mistakes in combining text chunks, which are located too close to each other, thus the input can be corrupted. Nurminen [9] in his thesis describes comprehensive PDF table detection and structure recognition algorithms that have demonstrated high recall and precision on "ICDAR 2013 Table Competition" [4]. Some of them are implemented in Tabula[2] system.

In contrast to the existing methods, we suggest a configurable system that is formulated as a set of customizable parameters and ad-hoc heuristics for recovering table cells from text chunks and rulings. We exploit features of table presentations in untagged PDF documents. Most of them such as horizontal and vertical distances, fonts, and rulings are well known and used in the existing methods. Additionally, we propose to exploit the feature of appearance of text printing instruction in PDF files.

Usually, when a table printed in a PDF document originally was an object (e.g. a table in a Word-document) then 1) one printing instruction forms a part or a whole of textual content of only one

[1]http://pdftohtml.sourceforge.net
[2]http://tabula.technology

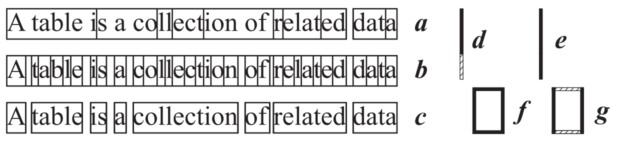
A table is a collection of related data
a
A table is a collection of related data
b
A table is a collection of related data
c
d
e
f
g

Figure 1: Preprocessing of text chunks (*a–c*) and rulings (*d–g*).

physical cell; 2) printing instructions forming a text inside each physical cell appear in the PDF file in the order that coincides with the human reading order of this text. We notice that it is true for many PDF generators. This feature can be especially useful in case of multi-row cells in table heads without rulings.

We also consider two configurations for the system and demonstrate experimental results based on the existing competition dataset, "ICDAR 2013 Table Competition", for both of them.

2. TABLE STRUCTURE RECOGNITION

We present the process of table structure recognition as three consecutive steps:

1. *preprocessing*: generating and preparing text chunks and rulings from a source document;

2. *text block recovering*: combining text chunks into text blocks;

3. *cell recovering*: dividing table space into rows, columns, and cells via text blocks.

2.1 Preprocessing

We operate two kind of objects: *text chunks* and *rulings*. A *text chunk* is defined as $c = (b, f, o, w)$, where

- $b = (x_l, y_t, x_r, y_b)$ — bonding box with four coordinates: $x_l = x_l(c)$ — left, $y_t = y_t(c)$ — top, $x_r = x_r(c)$ — right, and $y_b = y_b(c)$ — bottom, $x_l, y_t, x_r, y_b \in \mathbb{R}$, the x-coordinate increases from left to right, and y-coordinate increases from top to bottom;

- $f = (f_f, f_s, f_b, f_i)$ — font with the attributes: $f_f = f_f(c)$ — family (string value), $f_s = f_s(c)$ — size in points, $f_b = f_b(c)$: $f_b \in \{\text{true}, \text{false}\}$ — bold or not, $f_i = f_i(c)$: $f_i \in \{\text{true}, \text{false}\}$ — italic or not;

- $o = o(c) : o \in \mathbb{N}$ — index number in the order of the appearance of text chunks in the source PDF file.

- $w = w(c) : w \in \mathbb{R}$ — space width.

Initially each text chunk corresponds to one instruction of text printing. The same text can be presented in PDF by different printing instructions, depending on the used PDF generator, as shown in Fig. 1, *a–c*. At first, we split all text chunks (Fig. 1, *a*) into one-character chunks (Fig. 1, *b*) and merge them into word chunks with removing space characters and reindexing the order of their appearance (Fig. 1, *c*).

On this stage our system enables applying two ad-hoc heuristics for eliminating some kinds of "insular" text chunks from the further processing:

- H_1, *eliminating itemization text chunks*: if a text chunk contains only one character marking itemized lists (e. g. bullet, square), then it is excluded;

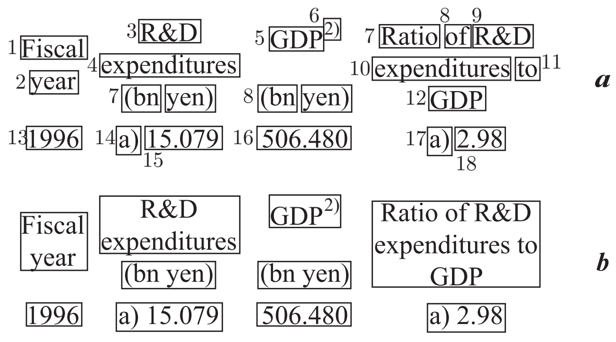

Figure 2: Text chunks and the order of their appearance (*a*), and text blocks constructed from them (*b*).

- H_2, *eliminating padding text chunks*: if a text chunk consists only of a series of padding characters (e. g. series of dots), then it is excluded.

Often, the two kinds of text chunks are visually detached from the rest of text chunks by long spaces. This lead to improperly recovered columns. Thus, eliminating them, we try to prevent some errors.

A *ruling* is defined as a bonding box with four coordinates: $r = (x_l, x_r, y_t, y_b)$. Visual rulings can be originally presented by printing instructions for lines and rectangles. We merge all segments of one visual line (Fig. 1, *d*) into one ruling (Fig. 1, *e*). We also split each rectangle (Fig. 1, *f*) into four rulings corresponding to its boundaries (Fig. 1, *g*).

2.2 Text Block Recovering

We define a *text block* as a set of chunks. On this step all text chunks are combined into blocks (Fig. 2). One chunk can be included only in one text block.

Text chunks are handled in pairs. We make a decision for each pair of chunks: to combine them or not. Two text chunks can be combined into one block when they satisfy the following conditions, in case of horizontal combining:

- P_1, *word spacing*: the horizontal distance between the chunks is less than a configurable threshold;

- P_2, *vertical projections*: there is a configurable intersection of their vertical projections;

or in case of vertical combining:

- P_3, *line spacing*: the vertical distance between the chunks is less than a configurable threshold;

- P_4, *horizontal projections*: there is a configurable intersection of their horizontal projections.

Moreover, a configuration can specify that two combining chunks c_1 and c_2 have to satisfy some or all of the ad-hoc heuristics listed below:

- H_3, *adjacency in the order of the appearance*: they are adjacent in the order of their appearance in the source PDF file, $o(c_1) = o(c_2) + 1$;

- H_4, *no rulings in text blocks*: there are no rulings in the rectangle between the chunks defined as

$$b(c_1, c_2) = \big(x_l(c_1, c_2), y_t(c_1, c_2), x_r(c_1, c_2), y_b(c_1, c_2)\big),$$

where

- $x_l(c_1,c_2) = \min(x_l(c_1), x_l(c_2))$,
- $y_t(c_1,c_2) = \min(y_t(c_1), y_t(c_2))$,
- $x_r(c_1,c_2) = \max(x_r(c_1), x_r(c_2))$,
- $y_b(c_1,c_2) = \max(y_b(c_1), y_b(c_2))$;

- H_5, *identical font family*: $f_f(c_1) = f_f(c_2)$;
- H_6, *identical font size*: $f_s(c_1) = f_s(c_2)$;
- H_7, *identical font bold attribute*: $f_b(c_1) = f_b(c_2)$;
- H_8, *identical font italic attribute*: $f_i(c_1) = f_i(c_2)$.

We suppose that each text block is a textual content of one cell, and each non-empty cell contains only one block. Thus, we try to recover non-empty cells without their arrangement in rows and columns.

2.3 Cell Recovering

In this step we construct rows and columns that constitute an arrangement of cells. The system provides two algorithms for slicing a table space into rows and columns. A configuration can use one of them.

The first (A_1) is based on the whitespace analysis. We use the algorithm [14] to recover horizontal and vertical gaps between text blocks. Each whitespace gap corresponds to a ruling. Thus, we try to recover all rulings, which separate cells in a table.

The second (A_2) is the analysis of connected text blocks. To generate columns, we first exclude each multi-column text block located in more than one column. We decide that a text block is multi-column when its horizontal projection intersects with the projections of two or more text blocks located in the same line. Each column is considered as an intersection of horizontal projections of one-column text blocks. Similarly, rows are constructed from vertical projections of one-row text blocks.

In this step we also recover empty cells. Some of them can be erroneous, i. e. they absent in the source table. The system provides the ad-hoc heuristic to dispose of erroneous empty cells:

- H_9, *cell singleton*: if a column contains only one non-empty cell then the column is merged with the nearest column to the left.

3. TWO CONFIGURATIONS

In the paper, we consider two configurations for our system. The main difference between them consists in estimation of word (P_1) and line (P_3) spacing, as well as used algorithm for cell construction.

The first C_1-configuration is the following settings:

- P_1, word spacing: $s_w = w * k_w$ where w is a space width of the left chunk, and k_w: $k_w \in \mathbb{R}$, $k_w > 0$ is a width factor;
- P_2, vertical projections: $y_t(c_1) \leq y_t(c_2) \leq y_b(c_1)$ or $y_t(c_1) \leq y_b(c_2) \leq y_b(c_1)$.
- P_3, line spacing: $s_l = h * k_h$, where h is a height of the upper chunk, and k_h: $k_h > 0$ is a height factor;
- P_4, horizontal projections: $x_l(c_1) \leq x_l(c_2) \leq x_r(c_2)$ or $x_l(c_1) \leq x_r(c_2) \leq x_r(c_2)$.
- Cell constructing: A_1-algorithm (whitespace analysis).
- Ad-hoc heuristics: H_1, H_3–H_9;
- Default values: $k_w = 1$ and $k_h = 1$.

The second C_2-configuration consists of the following settings:

- P_1, word spacing:
$$s_w = \begin{cases} w, & \text{if } w_{min} < |d| \leq w_{max} \\ w_{min}, & \text{if } |d| \leq w_{min} \\ w * k_w, & \text{otherwise;} \end{cases}$$
where w is a space width of the left chunk, k_w: $0 < k < 1$ is a width factor, d is the horizontal distance between the chunks, w_{min}: $t_1 \in \mathbb{R}$, $w_{min} > 0$ is a threshold (the minimum width of a space), w_{max}: $w_{max} \in \mathbb{R}$, $w_{max} > w_{min}$ is a threshold (the maximum width of the space);
- P_2, vertical projections: $y_b(c_1) = y_b(c_2)$;
- P_3, line spacing: $s_l = t_2$: $t_2 \in \mathbb{R}$, $t_2 > 0$ is a threshold;
- P_4, horizontal projections: $x_l(c_1) \leq x_l(c_2) \leq x_r(c_2)$ or $x_l(c_1) \leq x_r(c_2) \leq x_r(c_2)$;
- Cell constructing: A_2-algorithm (connected text block analysis);
- Ad-hoc heuristics: H_1–H_8;
- Default values: $k_w = 0.5$, $w_{min} = 4$, and $w_{max} = 56$.

4. EXPERIMENTAL EVALUATION

To evaluate both configurations we use the methodology for algorithms for table understanding in PDF documents proposed in the paper [3]. We also use the existing competition dataset[3], "ICDAR 2013 Table Competition" [4]. It contains 156 tables in 67 PDF documents collected from EU and US government websites.

The evaluated prototype of our system uses the iText[4] library for PDF interpretation to extract PDF objects from source documents and to generate the text chunks and rulings. In the evaluation, the parameters for both configurations have been set up by default values without searching for their optimal values. The experimental results are shown in Table 1. The highest F-score reaches more than 0.93.

Note that the evaluation was performed automatically using Nurminen's Python scripts[5] for comparing ground-truth and result files that implement this methodology with slight modifications. Therefore our results shown in Table 1 should not be matched directly with others demonstrated on "ICDAR 2013 Table Competition". Nevertheless, we can declare that the experimental results show the high performance of our system on the recognized dataset of PDF tables.

Table 1: Experimental results

Configuration	C_1	C_2
recall	0.9121	0.9233
precision	0.9180	0.9499
F-score	0.9150	0.9364

Moreover, we can improve F-score via setting optimal values for the configuration parameters. In both configurations, the numeric thresholds and factors can be set as the result of searching for maximum of F-score on a target dataset. For example, we have searched

[3]http://www.tamirhassan.com/dataset.html
[4]https://sourceforge.net/projects/itext
[5]http://tamirhassan.com/competition/dataset-tools.html

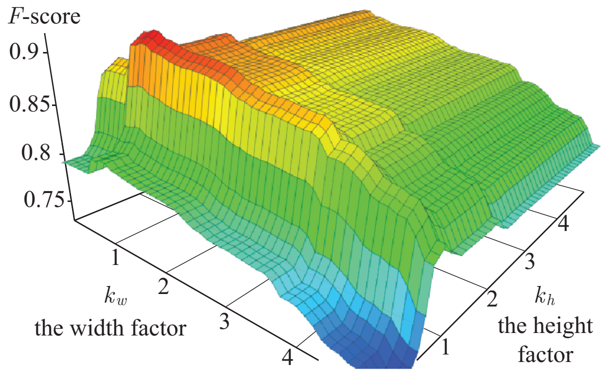

Figure 3: Searching for factor values to maximize F-score in C_1-configuration.

for the maximum of F-score as the function of two variables (the width and height factors) in the C_1-configuration on the competition dataset "ICDAR 2013 Table Competition" (Fig. 3). We have evaluated 2500 tests, where both k_w and k_h have increased from 0 to 5 with the step 0.1. The F-score have reached the maximum (0.9189) when the width factor k_w is 0.9 and the height factor is 1.0.

5. CONCLUSION AND FURTHER WORK

Unlike the existing solutions, our system enables the advanced configuration options which allow to adapt it to different sources. We have formulated a set of valuable ad-hoc heuristics that can be enhanced in the future. It is important to note, that it was for the first time, that we have examined the possibility of applying the order of the appearance of text chunks in PDF files for table structure recognition.

The main applications of our system are in the field of data accessibility, information extraction, and unstructured data integration. Particularly, we develop an experimental web-application[6] for PDF table extraction based on the prototype of our system. In the current state, this tool enables only manual table selection in a page of a PDF document and automatic table structure recognition. As the result of this process, an extracted table is accessible in the editable format, HTML or spreadsheet, that can be used as input in our rule-based spreadsheet data canonicalization system[7] for further transforming data from arbitrary tables to relational ones [13].

The further work is in progress on expanding the set of ad-hoc heuristics. We believe the involvement of the additional features such as text alignment, superscript, and subscript will allow to improve our system. We also expect an advancement in the preprocessing step for excluding "messy" rulings, which originate from underlined or striked text. In the future, our system also can be extended for supporting automatic PDF table detection.

6. ACKNOWLEDGMENTS

We thank Tamir Hassan for the detailed discussion and explanation of the methodology for evaluating algorithms for table understanding in PDF documents [3] in the part of table structure recognition. We also thank Anssi Nurminen for providing his Python scripts, which have allowed us to automate the evaluation process.

This work was financially supported by the Russian Foundation for Basic Research (grants 15-37-20042, 14-07-00166) and Council for Grants of the President of Russian Federation (grant NSh-8081.2016.9). Our web-application for PDF table extraction is performed on resources of the Shared Equipment Center of Integrated Information and Computing Network for Irkutsk Research and Educational Complex[8].

[6]available at http://cells.icc.ru/pdfte
[7]available at http://cells.icc.ru/ssdc

7. REFERENCES

[1] B. Coüasnon and A. Lemaitre. *Handbook of Document Image Processing and Recognition*, chapter Recognition of Tables and Forms, pages 647–677. Springer London, 2014.

[2] A. C. e Silva, A. M. Jorge, and L. Torgo. Design of an end-to-end method to extract information from tables. *International Journal of Document Analysis and Recognition (IJDAR)*, 8(2):144–171, 2006.

[3] M. Göbel, T. Hassan, E. Oro, and G. Orsi. A methodology for evaluating algorithms for table understanding in PDF documents. In *Proc. of the 2012 ACM Symposium on Document Engineering*, pages 45–48, New York, NY, USA, 2012.

[4] M. Göbel, T. Hassan, E. Oro, and G. Orsi. ICDAR 2013 table competition. In *Proc. of the 12th Int. Conf. on Document Analysis and Recognition*, pages 1449–1453, 2013.

[5] T. Hassan and R. Baumgartner. Table recognition and understanding from PDF files. In *Proc. of the 9th Int. Conf. on Document Analysis and Recognition - Volume 02*, pages 1143–1147, Washington, DC, USA, 2007. IEEE Comp. Soc.

[6] J. Hu and Y. Liu. *Analysis of Documents Born Digital*, pages 775–804. Springer London, London, 2014.

[7] S. Khusro, A. Latif, and I. Ullah. On methods and tools of table detection, extraction and annotation in PDF documents. *J. Inf. Sci.*, 41(1):41–57, Feb. 2015.

[8] Y. Liu, K. Bai, P. Mitra, and C. L. Giles. TableSeer: Automatic table metadata extraction and searching in digital libraries. In *Proc. of the 7th ACM/IEEE Joint Conf. on Digital Libraries*, pages 91–100, 2007.

[9] A. Nurminen. Algorithmic extraction of data in tables in PDF documents. Master's thesis, Tampere University of Technology, Tampere, Finland, 2013.

[10] E. Oro and M. Ruffolo. PDF-TREX: An approach for recognizing and extracting tables from PDF documents. In *Proc. of the 10th Int. Conf. on Document Analysis and Recognition*, pages 906–910, 2009.

[11] J. Y. Ramel, M. Crucianu, N. Vincent, and C. Faure. Detection, extraction and representation of tables. In *Proc. of the 7th Int. Conf. on Document Analysis and Recognition*, pages 374–378 vol.1, 2003.

[12] R. Rastan, H.-Y. Paik, and J. Shepherd. Texus: A task-based approach for table extraction and understanding. In *Proc. of the 2015 ACM Symposium on Document Engineering*, pages 25–34, 2015.

[13] A. Shigarov. Table understanding using a rule engine. *Expert Systems with Applications*, 42(2):929–937, 2015.

[14] A. Shigarov and R. Fedorov. Simple algorithm page layout analysis. *Pattern Recognition and Image Analysis*, 21(2):324–327, 2011.

[15] B. Yildiz, K. Kaiser, and S. Miksch. pdf2table: A method to extract table information from PDF files. In *Proc. of the 2nd Indian Int. Conf. on Artificial Intelligence, Pune, India*, pages 1773–1785, 2005.

[8]http://net.isc.irk.ru