# PyTabby: a Docreader's module for extracting text and tables from PDF with a text layer

Andrey A. Mikhailov[1], Alexey Shigarov[1] and Ilya S. Kozlov[2]

[1]*Matrosov Institute for System Dynamics and Control Theory of Siberian Branch of Russian Academy of Sciences, Irkutsk, 664033, Russian Federation*

[2]*Ivannikov Institute for System Programming of Russian Academy of Sciences, 25 Alexander Solzhenitsyn St., Moscow, 109004, Russian Federation*

**Abstract**

This paper presents a complete solution for extraction of textual information and tables from PDF with a text layer. The presented solution consist of two parts: PyTabby is a tool for extracting text and tables from PDF with a complex background and layout, and Python wrapper module for Docreader tool. The PyTabby tool extracts text and tables from the low level representation of the PDF format. It enables employment of the additional information excluded in scanned documents and provides improvement of quality and performance compared with Optical Character Recognition (OCR) methods. The presented solution is incorporated into Docreader tool to parse PDF files with a text layer and is used as a part of the TALISMAN technology for social analytics.

**Keywords**

document structure analysis, PDF documents, document analysis,

## 1. Introduction

In the digital world, web content is the main value. Being useful information that provides business opportunities, web content is a valuable asset on the Internet. A large number of such documents and their properties makes them a valuable resource in data science and business intelligence applications. Usually, electronic documents are not accompanied by the semantics necessary for machine interpretation of their content as intended by their author. The information accumulated in them is often unstructured and not standardized. Analysis of these documents requires their preliminary transformation to a structured representation with a formal model.

The Ivannikov Institute for System Programming of Russian Academy of Sciences is developing a social media analysis technology called TALISMAN[1]. Unlike most existing solutions for social analytics, the TALISMAN technology was originally aimed at working with large amounts of data. The most promising open solutions from the stack of Big Data technologies are employed, such as: Apache Spark, GraphX, MLLib, etc.

*Information Technologies: Algorithms, Models, Systems (ITAMS), September 14, 2021, Irkutsk*

mikhailov@icc.ru (A. A. Mikhailov); shigarov@icc.ru (A. Shigarov); kozlov-ilya@ispras.ru (I. S. Kozlov)
http://td.icc.ru (A. Shigarov)
0000-0003-4057-4511 (A. A. Mikhailov); 0000-0001-5572-5349 (A. Shigarov); 0000-0002-0145-1159 (I. S. Kozlov)

[1]https://www.ispras.ru/en/technologies/talisman

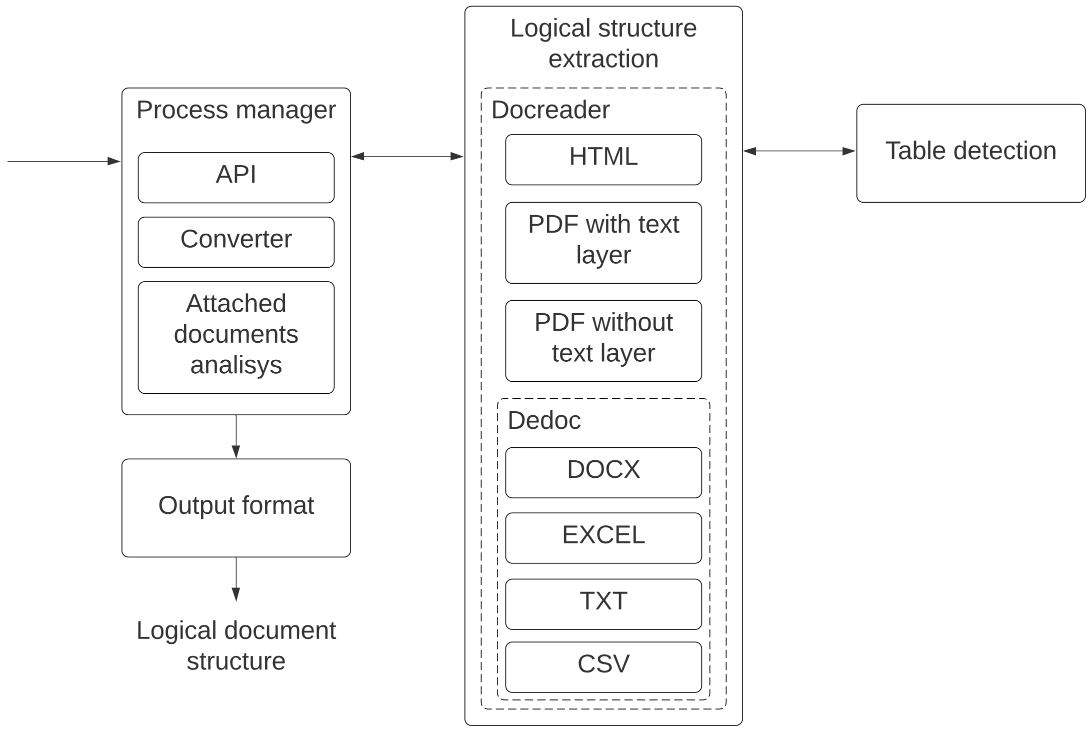


**Figure 1:** The Docreader scheme of work.

The part of TALISMAN is Docreader[2], a universal and open system for bringing documents to a single format. It automatically extracts the logical structure, tables, and meta-information. The content of the documents is presented in the form of a tree encoding headers and lists of various levels of nesting. Docreader can be integrated as a separate component into systems for analyzing the structure and content of documents.

Docreader can extract logical structure from PDF with and without a text layer. However, for table detection and recognition, it uses the OCR [1] for two cases.

## 2. Related research

In the past two decades, several methods and tools for PDF table extraction have been proposed. Some of them are discussed in the recent surveys [2, 3, 4, 5]. Ramel et al. [6] consider two techniques for detecting and recognizing tables from documents in an exchange format like PDF. The first technique is based on the analysis of ruling lines. The second analyzes the arrangement of text components. Hassan et al. [7] expand these ideas for the PDF table extraction. In the project TableSeer, Liu et al. [8] propose methods for detecting tables in PDF documents and

[2]https://education.at.ispras.ru/dedoc

extracting metadata (headers). They use text arrangement, fonts, whitespace, and keywords (e. g. "Table", "Figure"). Oro et al. [9] present PDF-TREX, a heuristic method where the PDF table extraction is realized as building from content elements to tables in a bottom-up way.

Yildiz et al. [10] propose a heuristic method for the PDF table extraction using pdftohtml[3] for generating its input. 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 [11] in his thesis describes comprehensive PDF table detection and structure recognition algorithms that have demonstrated high recall and precision on "ICDAR 2013 Table Competition" [12]. Some of them are implemented in Tabula[4], a tool for extracting tabular data from PDF. Rastan et al. [13] consider a framework for the end-to-end table processing including the task of table structure recognition. Moreover, Rastan et al. [14] suggest using an ad-hoc document analysis leading to a better table extraction. Their wrapper is able to detect features such as page columns, bullets, and numbering. Perez-Arriaga et al. [15] combines layout heuristics with a supervised machine learning method based on k-nearest neighbors to extract tables from untagged PDF documents. Their system, TAO, promises to be an efficient, comprehensive and robust solution for both stages: table detection and cell structure recognition, and it does not depend on fixed patterns or layouts of tables or documents.

## 3. The proposed method

The process of PDF table and textual information extraction involves the following phases [16]:

1. data preparation, to recover text blocks presented as words and ruling lines from instructions of a source PDF document;
2. text line and paragraph extraction, to recover text blocks presented as lines and paragraphs;
3. table detection, to recover a bounding box of each table located on a page;
4. table structure recognition, to recover a cell structure of a detected table.

We propose to use an heuristic-based page layout analysis to recover text blocks such as paragraphs, titles, footnotes, table cells etc. These additional data allow us to correct some errors of the presented table detection.

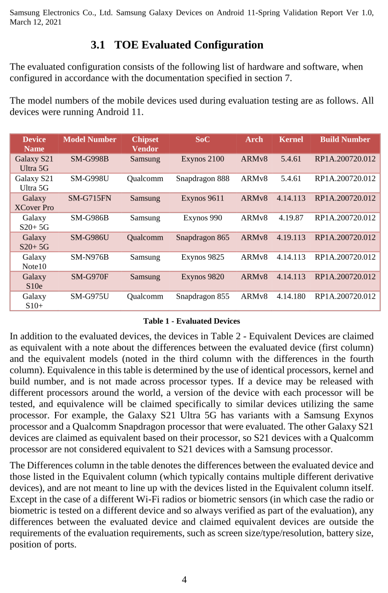
To build text blocks, we use data that are available in untagged PDF documents, including character positions, fonts, rulings, and cursor traces. Since such documents do not contain word structures, we propose a simple algorithm for combining adjacent character positions into words. Moreover, we adapt and extend the existing algorithms of T-Recs systems [17, 18] for combining neighbor words into text blocks. Originally, T-Recs algorithms were designed for document images. In contrast to them, our adoption uses additional heuristics based on the PDF-specific data.

[3]http://pdftohtml.sourceforge.net
[4]http://tabula.technology

## 3.1. Table detection

At this step, we detect only full and partial bordered tables. The main idea is finding table boxes on the page using ruling lines (vertical and horizontal). This can be done in two ways: based on the image and on the PDF instruction analysis. Both approaches have their disadvantages. In the first case, the image of the document contains redundant information, such as text, pictures, forms, etc. This information makes it difficult to highlight vertical and horizontal lines in the document image. The PDF format allows separate selection of all the instructions with which the outline of the table is formed - drawing lines, rectangles, etc. However, PDF printers often use non-standard approaches to output graphics. For example, the color of the line can be the same as the background, so that the visual line is invisible. Such a reading can be automatically processed, but if the color of the line and the background differs only slightly, it will be impossible to distinguish them, and, hence, their programmatic separation will also be difficult.

In this work, we use a combined approach. At the first stage, all instructions for outputting text and graphic information were removed from the PDF document, as shown in Fig 3.1. As a result, the document consisted only of horizontal and vertical lines. After such processing, an image was generated from the PDF document page by page. Using the algorithm of connected components, such an image can be used to detect the contours of the tables.



Samsung Electronics Co., Ltd. Samsung Galaxy Devices on Android 11-Spring Validation Report Ver 1.0, March 12, 2021

3.1 TOE Evaluated Configuration

The evaluated configuration consists of the following list of hardware and software, when configured in accordance with the documentation specified in section 7.

The model numbers of the mobile devices used during evaluation testing are as follows. All devices were running Android 11.

| Device Name | Model Number | Chipset Vendor | SoC | Arch | Kernel | Build Number |
|---|---|---|---|---|---|---|
| Galaxy S21 Ultra 5G | SM-G998B | Samsung | Exynos 2100 | ARMv8 | 5.4.61 | RP1A.200720.012 |
| Galaxy S21 Ultra 5G | SM-G998U | Qualcomm | Snapdragon 888 | ARMv8 | 5.4.61 | RP1A.200720.012 |
| Galaxy XCover Pro | SM-G715FN | Samsung | Exynos 9611 | ARMv8 | 4.14.113 | RP1A.200720.012 |
| Galaxy S20+ 5G | SM-G986B | Samsung | Exynos 990 | ARMv8 | 4.19.87 | RP1A.200720.012 |
| Galaxy S20+ 5G | SM-G986U | Qualcomm | Snapdragon 865 | ARMv8 | 4.19.113 | RP1A.200720.012 |
| Galaxy Note10 | SM-N976B | Samsung | Exynos 9825 | ARMv8 | 4.14.113 | RP1A.200720.012 |
| Galaxy S10e | SM-G970F | Samsung | Exynos 9820 | ARMv8 | 4.14.113 | RP1A.200720.012 |
| Galaxy S10+ | SM-G975U | Qualcomm | Snapdragon 855 | ARMv8 | 4.14.180 | RP1A.200720.012 |

Table 1 - Evaluated Devices

In addition to the evaluated devices, the devices in Table 2 - Equivalent Devices are claimed as equivalent with a note about the differences between the evaluated device (first column) and the equivalent models (noted in the third column with the differences in the fourth column). Equivalence in this table is determined by the use of identical processors, kernel and build number, and is not made across processor types. If a device may be released with different processors around the world, a version of the device with each processor will be tested, and equivalence will be claimed specifically to similar devices utilizing the same processor. For example, the Galaxy S21 Ultra 5G has variants with a Samsung Exynos processor and a Qualcomm Snapdragon processor that were evaluated. The other Galaxy S21 devices are claimed as equivalent based on their processor, so S21 devices with a Qualcomm processor are not considered equivalent to S21 devices with a Samsung processor.

The Differences column in the table denotes the differences between the evaluated device and those listed in the Equivalent column (which typically contains multiple different derivative devices), and are not meant to line up with the devices listed in the Equivalent column itself. Except in the case of a different Wi-Fi radios or biometric sensors (in which case the radio or biometric is tested on a different device and so always verified as part of the evaluation), any differences between the evaluated device and claimed equivalent devices are outside the requirements of the evaluation requirements, such as screen size/type/resolution, battery size, position of ports.

4

**Figure 2:** Original PDF page

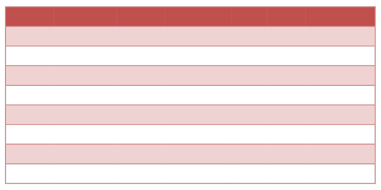


**Figure 3:** Cleared PDF page

### 3.2. Table structure recognition

At this step, we construct rows and columns that constitute an arrangement of cells. The system provides an algorithm for slicing a table space into rows and columns based on the analysis of connected text blocks. To generate columns, first, we exclude each multi-column text block located in more than one column. We assume 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. At this step, we also recover empty cells. Some of them can be erroneous, i.e. they are absent in the source table. The system provides the ad-hoc heuristics to dispose of erroneous empty cells.

## 4. Conclusion

In this paper, we develop PyTabby, a tool for the PDF table and textual extraction from PDF documents with a text layer. This extends our previous work for the table structure recognition [19, 20]. The system exploits a set of customizable ad-hoc heuristics for table detection and cell structure reconstruction based on features of text and ruling lines presented in 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 and positions of a drawing cursor.

## Acknowledgments

The results were obtained within the framework of the State Assignment of the Ministry of Education and Science of the Russian Federation for the project ”Methods and technologies of cloud-based service-oriented platform for collecting, storing and processing large volumes of multi-format interdisciplinary data and knowledge based upon the use of artificial intelligence, model-guided approach and machine learning” (State Registration No. 121030500071-2).

## References

[1] C. C. Tappert, C. Y. Suen, T. Wakahara, The state of the art in online handwriting recognition, IEEE Transactions on pattern analysis and machine intelligence 12 (1990) 787–808.

[2] B. Coüasnon, A. Lemaitre, Handbook of Document Image Processing and Recognition, 2014, pp. 647–677. URL: http://dx.doi.org/10.1007/978-0-85729-859-1_20. doi:10.1007/978-0-85729-859-1_20.

[3] J. Hu, Y. Liu, Analysis of Documents Born Digital, 2014, pp. 775–804. URL: http://dx.doi.org/10.1007/978-0-85729-859-1_26. doi:10.1007/978-0-85729-859-1_26.

[4] S. Khusro, A. Latif, I. Ullah, On methods and tools of table detection, extraction and annotation in PDF documents, J. Inf. Sci. 41 (2015) 41–57. URL: http://dx.doi.org/10.1177/0165551514551903. doi:10.1177/0165551514551903.

[5] A. S. Corrêa, P.-O. Zander, Unleashing tabular content to open data: A survey on pdf table extraction methods and tools, in: Proc. 18th Int. Conf. on Digital Government Research, 2017, pp. 54–63. URL: http://doi.acm.org/10.1145/3085228.3085278. doi:10.1145/3085228.3085278.

[6] J. Y. Ramel, M. Crucianu, N. Vincent, C. Faure, Detection, extraction and representation of tables, in: Proc. 7th Int. Conf. on Document Analysis and Recognition, 2003, pp. 374–378 vol.1. doi:10.1109/ICDAR.2003.1227692.

[7] T. Hassan, R. Baumgartner, Table recognition and understanding from PDF files, in: Proc. 9th Int. Conf. on Document Analysis and Recognition - Vol. 02, 2007, pp. 1143–1147. URL: http://dl.acm.org/citation.cfm?id=1304596.1304833.

[8] Y. Liu, K. Bai, P. Mitra, C. L. Giles, TableSeer: Automatic table metadata extraction and searching in digital libraries, in: Proc. 7th ACM/IEEE Joint Conf. on Digital Libraries, 2007, pp. 91–100. URL: http://doi.acm.org/10.1145/1255175.1255193. doi:10.1145/1255175.1255193.

[9] E. Oro, M. Ruffolo, PDF-TREX: An approach for recognizing and extracting tables from PDF documents, in: Proc. 10th Int. Conf. on Document Analysis and Recognition, 2009, pp. 906–910. doi:10.1109/ICDAR.2009.12.

[10] B. Yildiz, K. Kaiser, S. Miksch, pdf2table: A method to extract table information from PDF files, in: Proc. 2nd Indian Int. Conf. on Artificial Intelligence, Pune, India, 2005, pp. 1773–1785.

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

[12] M. Göbel, T. Hassan, E. Oro, G. Orsi, ICDAR 2013 table competition, in: Proc. 12th Int. Conf. on Document Analysis and Recognition, 2013, pp. 1449–1453. doi:10.1109/ICDAR.2013.292.

[13] R. Rastan, H.-Y. Paik, J. Shepherd, Texus: A task-based approach for table extraction and understanding, in: Proc. 2015 ACM Symposium on Document Engineering, 2015, pp. 25–34. URL: http://doi.acm.org/10.1145/2682571.2797069. doi:10.1145/2682571.2797069.

[14] R. Rastan, H.-Y. Paik, J. Shepherd, A pdf wrapper for table processing, in: Proc. 2016 ACM Symposium on Document Engineering, 2016, pp. 115–118. URL: http://doi.acm.org/10.1145/2960811.2967162. doi:10.1145/2960811.2967162.

[15] M. O. Perez-Arriaga, T. Estrada, S. Abad-Mota, TAO: system for table detection and extraction from PDF documents, in: Proc. 29th Int. Florida Artificial Intelligence Research Society Conference, 2016, pp. 591–596.

[16] A. Shigarov, A. Altaev, A. Mikhailov, V. Paramonov, E. Cherkashin, Tabbypdf: Web-based system for pdf table extraction, in: R. Damaševičius, G. Vasiljevienė (Eds.), Information and Software Technologies, Springer International Publishing, Cham, 2018, pp. 257–269.

[17] T. Kieninger, A. Dengel, The t-recs table recognition and analysis system, in: Document Analysis Systems: Theory and Practice, 1999, pp. 255–270. doi:10.1007/3-540-48172-9_21.

[18] T. Kieninger, A. Dengel, Applying the t-recs table recognition system to the business letter domain, in: Proc. 6th Int. Conf. on Document Analysis and Recognition, 2001, pp. 518–522. doi:10.1109/ICDAR.2001.953843.

[19] A. Shigarov, A. Mikhailov, V. Khristyuk, V. Paramonov, Software development for rule-based spreadsheet data extraction and transformation, 2019, pp. 1132–1137. doi:10.23919/MIPRO.2019.8756829, cited By 1.

[20] A. Shigarov, I. Cherepanov, E. Cherkashin, N. Dorodnykh, V. Khristyuk, A. Mikhailov, V. Paramonov, E. Rozhkow, A. Yurin, Towards end-to-end transformation of arbitrary tables from untagged portable documents (pdf) to linked data, volume 2463, 2019, pp. 1–12. Cited By 0.