Klemens Sudi, BSc

TabEx - Multi-Approach Table Extraction for Native and Scanned PDF Documents

Master's Thesis

to achieve the university degree of

Master of Science

Master's degree programme: Software Engineering and Management

submitted to

Graz University of Technology

Supervisor

Dipl.-Ing. Dr.techn. Roman Kern

Institute for Interactive Systems and Data Science
Head: Univ.-Prof. Dipl.-Inf. Dr. Stefanie Lindstaedt

Graz, September 2019

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master‘s thesis.

Date	Signature

Abstract

The subject area of automated Information Extraction from PDF documents is of high relevance since the PDF standard is still one of the most popular document formats for information representation and exchange. There is no structuring blueprint for PDF documents, which makes automated information gathering a complex task. Since tables are structuring elements with a very high information density, the field of Table Detection is highly relevant in the context of Information Extraction. Due to the high variety of formats and layouts it is hard to choose the correct tool that suits optimally for every specific scenario. In this thesis, the added value of techniques used to identify table structures in scanned PDF documents is evaluated. Therefore, two algorithms were implemented to allow an objective comparison of Table Extraction applied on different types of PDF documents. While the algorithm developed to treat native PDFs is based on heuristics, the second approach relies on deep-learning techniques. The evaluation of both implementations showed that the heuristic approach performs excellent in detecting tables. However, it shows weaknesses in distinguishing non-tabular areas that show similarities to table structures, from tabular areas. Therefore, the Recall metric shows better results than the Precision for the heuristic method. When applying Table Detection on scanned PDFs using the second approach, the low number of False Positives and therefore the superior Precision value compared to the first approach is notable. On the other hand the number of tables not detected as trade-off for the high Precision result in a lower Recall for single- as well as multi-column documents if partial detections are classified as correct results. Furthermore, limitations that reduce the detection-ratio were detected. This concerns structures that share similarities with tables, like figures, formulas and pseudo-code. These mentioned limitations are particularly relevant for the heuristic and less for the deep-learning based approach. All in all, there were several findings concerning advantages and disadvantages of applying Table Detection on scanned and native documents. Based on the evaluation results, strategies were elaborated of when to preferably use a specific approach dependent upon the document type, layout and structuring elements.

Contents

Abstract iii

1 Introduction 1

2 Related Work 3
2.1 Background . . . 3
2.1.1 Types of PDFs . . . 4
2.1.2 Text Extraction of PDFs . . . 6
2.1.3 Optical Character Recognition . . . 8
2.1.4 Tables in PDFs . . . 12
2.1.5 Deep Learning . . . 15
2.2 State of the Art and Trends . . . 18
2.2.1 Native PDF Table Extraction . . . 18
2.2.2 Scanned PDF Table Extraction . . . 22
2.2.3 Overview of State of the Art Table Detection and Extraction Tools . . . 26

3 Method 27
3.1 Native PDF Table Extraction . . . 28
3.1.1 Technologies . . . 29
3.1.2 Implementation . . . 29
3.1.3 Problems and Challenges . . . 48
3.2 Scanned Table Extraction . . . 51
3.2.1 Technologies . . . 52
3.2.2 Implementation . . . 53
3.2.3 Problems and Challenges . . . 62

4 Evaluation 65
4.1 Evaluation Criteria . . . 65

4.2 Limitations 70
4.3 Test Data 71
4.3.1 Native Test Data 72
4.3.2 Scanned Test Data 73
4.4 Results 74
4.4.1 Native PDF Table Detection 74
4.4.2 Scanned PDF Table Detection 77
4.4.3 Comparison of Approaches 79
4.5 Discussion 80

5 Conclusion **83**
5.1 Future Work 85

Bibliography **87**

List of Figures

2.1 Example native PDF 5
2.2 Example scanned PDF 6
2.3 Different Types of Tables 14

3.1 The process of Table Extraction from high-level perspective. . . . 28
3.2 Preprocessing of native PDFs 30
3.3 Summarization of words from native PDF 31
3.4 Summarization of lines from native PDF 32
3.5 Table Detection of native PDFs 33
3.6 Sparse Line Detection 34
3.7 Example of intended sparse line 35
3.8 Multi column detection example two columns 37
3.9 Multi column detection example one column 37
3.10 Handling of bullet points and justified text 38
3.11 Header and Footer Detection based on Keywords 42
3.12 Header and Footer Detection based on Line Distance 43
3.13 Table Extraction of native PDFs 44
3.14 Example summarized lines because of y-distance 46
3.15 Critical column boundary detection 47
3.16 Training Process for scanned Table Detection 54
3.17 Ground Truth of scanned PDF Tables 55
3.18 Transformation of scanned PDF Page 56
3.19 Training process with TensorFlow based tool Luminoth 57
3.20 Process for scanned Table Detection 58
3.21 Process for scanned Table Boundary Detection 60
3.22 Process for scanned Table Extraction 61
3.23 Table boundaries given coordinates for extraction of tabular data 62

4.1 Statistical parameters to calculate performance of Table Detection 66

4.2 Visualisation of Table Detection Results 69

List of Tables

2.1 Comparison of PDF text extraction tools 8
2.2 Overview of Table Extraction and Table Detection algorithms . . 26

4.1 Description of native test data 73
4.2 Description of scanned test data 74
4.3 Results of native Table Detection process 75
4.4 Native detection results (partial valid) 76
4.5 Native detection results (partial invalid) 76
4.6 Results of scanned Table Detection process 77
4.7 Scanned detection results (partial valid) 78
4.8 Scanned detection results (partial invalid) 78
4.9 Scanned and native detection results (partial valid) 79
4.10 Scanned and native detection results (partial invalid) 79

1 Introduction

Since the PDF (Portable Document Format) standard still is one of the most popular electronic documents formats, the extraction of text and additional information like tables or images from PDF documents is a relevant problem today. The extraction of tables, which is investigated on in this paper, is a very relevant task because of the high information density and the common usage as structuring element in documents. Since tables can be represented in a variety of formats, for example using lines or white spaces to point out cell boundaries, extracting information is a very challenging task [39]. There are countless tools available which try to solve this problem in different ways. Due to the high variety and missing documentation of these tools, their functionalities, as well as their quality, is hard to determine. This makes it difficult to choose the perfectly suited tool for a specific use case.

One of the challenges of performing text extraction and other techniques for collecting information from PDF documents is the fact that the PDF format is layout based. In other words, in the PDF standard, the fonts and positions of the individual characters are specified whereas the role of words or paragraphs in the documents, as well as their semantics, are usually not provided in the majority of cases. In the rare case a PDF document is tagged with semantic information the required quality to profit from them is rarely given [19].

Additionally, to extracting text and structural information from PDF documents, there are many use cases for which native PDF text extraction techniques are not suitable. This includes for example to extract text and tables from images. For these scenarios, Optical Character Recognition (OCR) techniques are highly suitable to collect information. OCR allows the recognition of characters through an optical mechanism which makes it possible to convert handwritten and typewritten text as well as scanned images of printed documents into machine-encoded text. Even though OCR is not able to compete with human reading skills, it makes it possible to convert certain types of data into editable and searchable data [38].

1 Introduction

Table Extraction (TE) is one subfield of Information Extraction (IE). For successful Table Extraction, the tables have to be detected and the cell-information has to be identified and extracted [48]. Since tables can be found in nearly every type of document, high quality TE will lead to a relevant increase in information content. As a cell of a table typically contains one unit of information, the extracted information of tables is highly suitable for automated machine processing [34].

Although the problem of Information Extraction from PDF tables is from high practical importance, there does not yet exist the ”perfect tool” for this subject area. The goal of this thesis is to develop a strategy of when to use classical Information Extraction and when to use OCR to generate optimal results concerning Table Extraction from PDF documents. To reach this goal, an evaluation pipeline will be defined in advance, which allows analysing the results of the different strategies used. To produce the results, tools using both strategies are developed and the used technologies, as well as algorithms, will be described. The results are evaluated and the optimal use cases for both strategies are outlined. The evaluation results will indicate if the usage of OCR additionally to classic PDF extraction techniques adds value concerning the detection and extraction of tabular content. Furthermore, the scenarios where both approaches can benefit from one another are explored and the findings will be documented and analyzed.

2 Related Work

The target of this chapter is to give the reader an overview about terms and concepts that are relevant for the further understanding of the thesis. Technical terminology is explained and an overview of the main underlying technologies is given in the ”Background” section. Furthermore, in the section ”State of the Art and Trends”, the current state of the problem targeted by the research question is examined based on several scientific research papers of the subject area.

2.1 Background

The Portable Document Format (PDF) is a file format used for visual presentation developed by Adobe in the 1990s with the goal to help users to exchange and view electronic document in an easy and reliable manner without having to consider the platform or environment in which the document was created or is looked at. The document format is based on a similar imaging model as PostScript, a page description language also published by Adobe, for device and resolution independent definition of text and graphics. The PDF standard defines a more structured format to enhance the performance compared to the most PostScript language programs. Additionally, object annotations and hyperlinks are supported to decrease the unfriendliness concerning interactive viewing and document interchange [36].

Adobe defines the PDF standard as follows: ”A PDF document consist of a collection of objects that together describe the appearance or one or more pages, possibly accompanied by additional interactive elements and higher-level application data. A PDF file contains the objects making up a PDF document along with associated structural information, all represented as a single self-contained sequence or bytes.” The appearance of every page of a PDF document is described by its content stream, which consists of a sequence of graphical objects. Adobe underlines, that PDF

documents can also include higher-level information like identification and logical structure information that help when searching and editing content. This factor also facilitates the process of content extraction for later reuse somewhere else. The technologically advanced imaging model allows pages to consist of text, graphical shapes and images in terms of abstract elements rather than in pixels. This way the high quality can be maintained on a wide range of devices (printers, displays). After the description of the desired output is generated by the page description language (a role of PDF), it is possible for applications to interpret and render the content independent of its type.

Even though two PDF documents may look the same on first glance, the method used for their creation can be completely different. PDFs can be generated directly from applications, which allows the use of the full capabilities of the PDF standard including tags, the imaging model and document interchange features. Applications that do not support direct PDF creation can generate them indirectly by calling the operating systems printing API. Also, third party translation software can be used for PDF document creation. While indirect PDF generation often is the easiest way of creating PDF document, not all high level structuring features of the standard can be used. The application specific information for document description is lost in this process and therefore not available to the driver or translating software [5].

2.1.1 Types of PDFs

PDF documents can be created either directly or indirectly by converting the content. When directly creating PDFs, the full capabilities of the standard can be used. The conversion of content into the PDF format is the easiest and most common way PDFs are obtained but generally the resulting documents are not taking advantage of the high-level Adobe imaging model by describing the output at very low level. After the PDF was created indirectly, all higher-level information is lost and can not be recovered anymore. This makes tasks in the context of data extraction and further processing a lot harder since most information has to be recovered on the basis of the incomplete information included in the document [36].

As mentioned in section 1, there are two main categories in which PDF documents can be categorized. Which category a PDF belongs to depends heavily on the way it was created.

The mentioned categories are [33, 41]:

- **Native PDF documents**
  So called native PDFs (also called true PDFs, Read PDFs or born digital PDFs) are originally created from a computer by software to generate reports, spreadsheets etc. and are made up of code that allows to present the content to the user in the same form the document was created originally. These vector-based files enable the viewer to search the file or perform copy-paste actions. Native PDFs can also be split into two sub categories like described above: Either PDFs are created directly by programs or indirectly by conversion from other file formats. These sub categories differ from one another by the amount of high-level information that is included. While directly created PDFs take advantage of all functionalities coming along with the usage of the PDF standard, indirectly created PDFs are the more common choice because of the simplicity of this approach [36, 41]. As shown in figure 2.1, due to its vector-based characteristic the quality of text remains the same when zooming into the PDF file.

Figure 2.1: Example text snippet of a native PDF. The edges stay sharp when zooming into native PDFs due to its vector based structure.

- **Scanned PDF documents (Image only)**
  Documents of this category do not have any digital code to describe its structure or formatting. These documents may have originally been native PDFs but by performing a scan they become no more than an image with the advantage that the file integrity can be maintained. Editing of scanned PDFs is very difficult and requires technical knowledge as well as appropriate tools. Adding tags to older or scanned PDFs to facilitate further processing is a very challenging and complex task since every PDF is designed and structured in its particular way. Working with so called ”image only” PDFs can be challenging since the missing text layer also prevents the user to perform searching and copying actions [41]. In contrast to figure 2.1, one can

recognize in figure 2.2 that zooming into scanned PDF documents lowers the quality of the textual elements because of its image like behaviour.

Figure 2.2: Example text snippet of a scanned PDF. The unsteady edges show the image-like behaviour of scanned PDF documents when zooming in.

Additionally to these two main categories, Korneev et al. [41] mentions **searchable PDF documents**. Searchable PDFs are positioned in between native and scanned PDFs since the picture of the scanned document is retained while the recognized text is placed behind the picture. When performing Optical Character Recognition (OCR), which will be described in detail in the following paragraphs, on native PDF documents so called searchable PDFs are created. Depending on the OCR software the quality of the textual layer can vary and optimal results of copying and searching actions cannot be guaranteed.

2.1.2 Text Extraction of PDFs

Bui at al. [21] define data extraction as a standard process in systematic review development. Since such processes usually rely on manual actions which are slow, costly and error prone, automated systems for text extraction and further processing are in high demand. Another article by Bui et al. [22] states that Information Extraction (IE) systems have a huge potential for helping humans performing time consuming extraction tasks. The downside is that most IE systems are not designed for data extraction from PDF typed documents. The reason why most IE tools are not focusing on the PDF standard, even though PDF documents are a very common and important extraction source, especially in the scientific domain, is the high complexity related to the implementation of such tools [22]. PDFs are confusing narrative content with meta-data as well as semi-structured text which is very hard to differentiate for automated algorithms. According to Damerow et al. [26] the conversion of images and PDF files into plain text is in constant demand since

further processing steps like Named Entity Recognition or topic modelling rely on textual content. The domain of scientific text mining is also mentioned to be relying on automated approaches for text extraction of articles [29]. Ferres et al. also mention that the PDF file format still represents the majority of scientific articles even though other data formats are emerging. After extracting textual information from PDF documents further processes like Natural Language Processing, Parsing, relation extraction and several others can be applied for enriching and mining textual content.

Bast and Korzen [19] describe the word identification process within PDF documents as non-trivial. They mention spacing as a main challenge. Whitespaces can vary a lot between letters and words and can also differ from line to line for example if the text is justified. Therefore, it is hard to define words by just considering the spacing distances. Other challenges can be line breaks within words and special characters which cannot be uniquely identified. Different styles of PDFs (single column, multiple columns) lead to problems concerning the correct identification of the word order. Determining paragraph boundaries is hard since these boundaries can be interrupted by figures, tables and page breaks. Keeping semantic information of PDF documents like titles and captions is an important task to allow further processing in more detail.

Many tools and APIs for performing automated text extraction from PDF documents are already available for the public but a big variance in quality and the number of features (like the identification of paragraph boundaries, the identification of the reading order or the identification of semantic roles) make it hard to evaluate these approaches. In 2017, Bast and Korzen [19] conducted a detailed analysis of multiple approaches and give an in-depth overview of the performance and result quality as well as the features every approach is providing. Following Table 2.1 is an excerpt or the provided results and gives an overview on the performance and quality of three of the evaluated text extraction libraries. Bast and Krozen analysed and evaluated the tools based on ten attributes during the extraction process.

This evaluation shows that concerning the most important criteria like missing and misspelled words all algorithms provide similar results. Concerning missing Newlines the tool PDFAct (former known as Icecite) performs the best by far because PDFBox does not provide automated paragraph boundary detection. PDFExtract performs badly in this category because isolated formulas are not categorized as single paragraphs. The high result within the spurious paragraph evaluation of

Criteria	PDFBox[1]	PDFExtract[2]	PDFAct[3] (Icecite)
Spurious Newines	3.6%	11.0%	4.0%
Missing Newlines	85.0%	40.0%	13.0%
Spurious Paragraphs	27.0%	21.0%	4.2%
Missing Paragrahps	0.2%	25.0%	5.5%
Reordered Paragraphs	0.1%	0.9%	0.1%
Spurious Words	0.5%	0.4%	0.3%
Missing Words	0.0%	0.1%	0.1%
Misspelled Words	1.6%	1.8%	0.6%
Extraction Error	2	176	34
Average Extraction Time/File	8.8 sec.	46 sec.	41 sec.

Table 2.1: Comparison of the PDF text extraction tools PDFBox, PDFExtract and PDFAct. The in-depth analysis of ten selected criteria allow a detailed evaluation of different areas concerning text extraction from PDF documents.

PDFBox is caused by the missing extraction of semantic roles. Even though PDFExtract take semantic roles into account, its performance is not significantly better while only PDFAct can deliver good results in this field. None of the three compared tools struggle with processing multi column documents (reordered paragraphs). The most obvious differences can be observed when comparing the results of the criteria extraction error and the average time needed for one PDF document. The PDFBox library has by far the lowest extraction error rate and also the extraction time per document is significantly faster compared to the other tools. To perform text extraction from scanned or image-typed documents, another technology called Optical Character Recognition is needed to pre-process the documents.

2.1.3 Optical Character Recognition

Cheung et al. [24] define Optical Character Recognition as "*the process of converting a raster image representation of a document into a format that a computer can process*". According to Islam et al. [37] OCR is defined as "*the process of digitizing a document image into its constituent characters*". Even tough research in this field has been performed for decades, the quality of OCR still stays behind human recognition skills. This area covers multiple subcategories of computer science like image and

natural language processing, pattern recognition, artificial intelligence and database systems [24].
To convert scanned PDF documents to searchable PDF files, Optical Character Recognition software is used. OCR tools recognize and extract data from the document pages which store the visual content of the original files in form of images. To differentiate between scanned and native PDFs one can check the document properties for font information that is not included in scanned files since PDF-Converters cannot recognize the text without performing OCR. This also means that only areas of images can be selected while native PDFs allow text selection for e.g. copying text chunks. Masum et al. [45] describe OCR as a very important technology in the context of automated analysis of business-related documents. It allows to archive digital copies of documents that cannot be manipulated in retrospect. This allows automated knowledge extraction to improve business efficiency by performing search operations based on domain specific key phrases.
As mentioned before, OCR technology enables the conversion of imaged typed files into machine encoded data that can be edited electronically, searched, digitally stored and used in machine processes. Therefore institutions like libraries commonly make use of the OCR technology to digitize their inventory. According to Chirag Patel et al [47], the accuracy of Optical Character recognition is heavily dependent on the text pre-processing and segmentation algorithms. Different styles, sizes, orientations of text and complex backgrounds make it a challenging task to detect letters in images. Characters with little visible difference (e.g. digit 0 and letter O) are very difficult to distinguish from one another and text embedded in very dark background can hardly be recognized by a computer. In 1870 CR Carey of Boston Massachusetts invented a retina scanner as image transmission system which was the origin for character recognition. This retina scanner was just able to characterize one font at a time and needed to be trained with pictures of every single character in advance. Generally speaking, OCR systems were very limited in the past by focusing on machine printed textual content and just small sets of handwritten text and symbols. In the majority of cases they were based on comparisons of images to a library of images, also called template matching [23].

In the year 1955 the first commercial OCR system for digitizing office documents was published. Nowadays many OCR tools of varying quality concerning their recognition accuracy are available on the market but only few of them are free to use and open source [47].

According to Chaudhuri et al. [23] Optical Character Recognition systems can be based on several different techniques such as: optical scanning, location segmentation, pre-processing, segmentation, representation, feature extraction, training and recognition and post-processing.

- **Optical scanning** converts multilevel images into bi-level black and white images. For this purpose, a threshold is defined where grey levels above this threshold are categorized as white and grey levels below are categorized as black. More sophisticated thresholding techniques take local properties of each file into account when defining the boundaries.
- **Location segmentation** is essential to locate regions which are textual and separate them from regions consisting of images before continuing with the recognition process. Most commonly every single character gets isolated and recognized individually without its context. Problems of segmentation techniques occur when characters touch their neighbours or when graphics are confused with text and vice versa. Another challenge is to get rid of the amount of noise which is contained in documents after they were scanned in and digitized as images.
- **Pre-processing** targets on producing data that is simple to analyse precisely for OCR systems. Pre-processing is split into three components: noise reduction, normalization of data and compression in the amount of information to be retained. Depending on the quality and resolution of the scanner as well as the physical document, the characters and lines can be smeared or broken. By using smoothing techniques like filling and thinning, small gaps in broken characters are closed and the width of lines is reduced. To receive standardized data by removing the variations of the writing, normalization techniques are applied. Thereby characters are adjusted to the same size, rotation and slant. When compressing the data to be retained after performing noise reduction and normalization, the shape information has to be maintained.
- **Segmentation** aims to divide characters into its sub-components. There are several different approaches for this problem but especially the segmentation

of cursive writing still is to be solved. By separating every line of a character from one another the recognition rate is affected directly.

- In which way the images are passed to the recognizer is handled in the **representation** step. The features which maximize the recognition rate are selected. This aims on maximizing the variability between classes by extraction the most representative information from the available data.
- **Feature extraction** is one of the most challenging problems of pattern recognition. Essential characteristics of symbols are captured for being able to accurately classify them afterwards. Frequently used techniques for feature extraction are template matching, distribution of points and structural analysis. The second part of this step focuses on classification which is the process of identifying each character and assign it to the matching class. This classification can be realized with the help of decision theoretic and structural methods. Especially when using structural methods, relationships between characteristics are highly relevant to classify characters correctly.
- The main approaches used for the **training and recognition** step are template matching, statistical techniques, structural techniques and ANNs. The approaches can be used standalone or as combined variations since they do not exclude each other. Each of these techniques use either holistic or analytic strategies for the completion of this step. While the holistic strategy makes use of top down approaches for full character recognition, the analytic strategy works bottom up and focuses on forming meaningful content starting on character level.
- **Post Processing** aims at grouping, detecting errors coming up in the former stages and correcting them. Grouping single characters to strings is based on their position in the document. Hereby the distances between characters and words are calculated and used as threshold for the group-assignment. Since no recognition systems can insure completely correct results, error detection by using the context is used to improve the outcome. Hereby probabilities of sequences of characters appearing together are analysed and utilized to detect errors. Also, the usage of dictionaries is a popular solution to efficiently detect and correct errors. Words not contained in the dictionary are classified as errors and corrected to the most similar existing word. On the one hand this process is very reliable and efficient but on the other hand this task is very time consuming.

2.1.4 Tables in PDFs

According to Göbel et al [32], the field of table understanding attracted much attention in the previous years, especially in the context of the PDF file format, since it is the most commonly used data format aside from HTML. The article splits the process of table understanding into the following three steps: Table Detection, table structure recognition and table interpretation. The biggest issue evaluating different table understanding approaches is the lack of documents where the ground truth is provided without having to pre-process the documents before the analysis.
In an article [25] published in the year 2017, the PDF standards report like characteristic that is focused on graphical visualization is described as a problem, since open data should be made open to the public in a way that that allows further automated machine processing. This circumstance has the result that many governmental data organizations are publishing their data in PDFs in addition to presenting structured data in .csv files, since they do not require as much effort as PDFs when including tables in a structured and comprehensible way. For these described reasons, the demand of tools for automated Table Extraction is given, especially in the sector of open government data communities, whose goal is to make governmental data easily accessible for the public in a structured way.
Even though a table is usually considered a simple concept, Embley et al. [28] state that defining a table in general is a difficult task. According to the free dictionary[4], a table is defined as follows: "*An orderly arrangement of data, especially one in which the data is arranged in columns and rows in an essentially rectangular form*". The Cambridge Dictionary[5] describes tables as "*an arrangement of facts and numbers in rows or blocks, especially in printed material*". Tables can contain different types of data (words, numbers, graphics) and can be generated in different ways (handwritten, electronically). Hassan et al. [34] state that data of table cells is very suitable for machine processing since one distinct logical item is contained.

Ranka et al. [49] describes Table Detection in scanned PDFs as an unsolved problem since there is no algorithm that is capable of identifying tables for all possible layouts. The biggest challenge is the variety of different table layouts which results in a problem of very high complexity. Complex tables can contain rows and columns spanning multiple areas and also cells not containing any value can exist. Figure 2.3 demonstrates the huge variety of different table layouts and types of tables.

[4] https://www.thefreedictionary.com/table
[5] https://dictionary.cambridge.org/dictionary/english/table

Tables are categorized into two main groups by Tran et al. [52]:

- **Ruling-line table**
  Tables of this type are separated from the content by bounding boxes or ruling lines. Also, separation lines within the table region are common. Ruling-line tables can further be categorized into: closed table (a rectangular boundary for table and table cells), non-closed table (no continuous rectangular boundary), parallel table (only parallel horizontal lines) and color table (structure through color blocks).
- **Non-ruling-line table**
  Non-ruling-line tables are common for reports, letters and emails but hardly used in scientific articles. Neither bounding boxes nor ruling-lines are used for content structuring.

Since a majority of PDF files do not include content meta-data in form of tags, it is hard to identify structural information. Additionally, noise resulting from pre-processing steps (like OCR) might complicate the correct identification of the document structure [44][53]. Especially in the domain of scientific publication, complex table structures with nested elements and columns and rows spanning multiple fields are very common. It is a big challenge for automated approaches within the field of Information Extraction that today most scientific articles are available as PDF lacking structural information [30]. Since tables do not have a unique characteristic that allows their reliable identification in documents, it is hard to teach computers to detect and understand tables even though they are easily recognizable for humans. A typical example is the way in which cell boundaries are defined. While some documents use lines as delimiters, other documents only use white spaces to separate cells and rows to achieve a table view [53].

Horizontal Separation Lines only

	Involvement of pupils in		
	Preperation and planing	Production of materials	Presentation and evaluation
Knowledge and awareness of different cultures	0,2885	0,3974	0,3904
Foreign language competence	0,3057	0,4184	0,3899
Social skills and abilities	0,3416	0,3369	0,4303
Acquaintance of special knowledge	0,2569	0,2909	0,3557
Self competence	0,3791	0,3320	0,4617

* Significance p = 0,000

No Separation Lines

	replies	%
individual person	540	(31.2%)
governmental body	141	(8.2%)
university/higher education	456	(26.4%)
commercial organisation (including consultancy) more than 250 employees	115	(6.7%)
commercial organisation (including consultancy) less than 250 employees	144	(8.3%)
association (e.g. trade association, trade union, employers association, chamber of commerce, NGO)	113	(6.5%)
other (please specify)	218	(12.6%)

Empty Cells

Country	BE	CZ	DK	DE	EE	EL	ES	FR	IE	IT	CY	LV	LT	LU	HU
Public	x	x				x	x	x	x	x		x	x	x	x
Private	x						x						x	*	x
Country	**MT**	**NL**	**AT**	**PL**	**PT**	**SI**	**SK**	**FI**	**SE**	**UK**	**IS**	**NO**	**BG**	**RO**	
Public	x	*		x	x		x	x		x			x	x	
Private	x	*			x			x		x					

Missing Cells

	0-4 years	6-7 years	8-10 years	11-16 years	> 16 years	TOTAL
Number of responses	1.528	1.058	729	787	2.008	6.110
Weights	25,01%	17,32%	11,93%	12,88%	32,86%	

Separation between Content and Headlines

Grade	1	2	3	4	5	6	7	8
A	13,1%	11,0%	6,8%	5,7%	5,5%	5,2%	5,2%	4,9%
B	11,9%	10,5%	6,4%	4,9%	4,8%	4,7%	4,5%	4,3%
C	8,5%	6,3%	4,6%	4,0%	3,9%	3,7%	3,6%	3,5%
D	6,1%	4,6%	4,3%	4,1%	4,0%	3,9%	3,7%	3,6%

Spanning Rows and Columns

Accuracy of the replies obtained from the survey carried out by CARSA (2006, Total survey population N=6190)				
Country	Valid replies per country	Number of researchers per country (1)	d (2)	Accuracy Level
Latvia	7	8.002	0,37027	
Malta	18	975	0,22896	
Iceland	18	5.466	0,23063	Low
Luxembourg	22	4.135	0,20841	
Israel	25	30.700	0,19592	

Complex Structure with Spanning Rows and Columns

Spain						
Scientific domain	Total Yearly Salary Costs of Researchers (data from the study)		Equivalent(s) profession(s) defined by ISCO classification	Category with available data in Eurostat	Total Yearly Salary Costs of similar professions (data from Eurostat)	
	Male	Female			Male	Female
Social and Human Sciences	27.301 €	16.806 €	2.4.4	2.4	46.657 €	30.443 €
Economics	42.978 €	33.669 €	2.4.1	2.4	46.657 €	30.443 €
Chemistry	39.091 €	20.296 €	2.1.1	2.1	42.138 €	31.056 €
Physics	27.197 €	16.067 €	2.1.1	2.1	42.138 €	31.056 €
Life Sciences	36.523 €	22.139 €	2.2.1	2.2	37.111 €	28.705 €
Mathematics	45.389 €	41.107 €	2.1.2	2.1	42.138 €	31.056 €
Information Sciences	26.059 €	25.893 €	2.1.3	2.1	42.138 €	31.056 €
Engineering Sciences	34.316 €	25.435 €	2.1.4	2.1	42.138 €	31.056 €
Environment and Geosciences	29.210 €	14.847 €	2.1.4	2.1	42.138 €	31.056 €

Figure 2.3: This figure shows some of the most common types of tables presentation. The row and column separation as well as the filling of the cells may vary heavily and makes automated Table Detection and extraction a very challenging task.

2.1.5 Deep Learning

For this thesis it is not only necessary to understand the nature of PDF documents and tabular structures, but also a fundamental knowledge concerning the field of Deep Learning is essential since many Table Extraction approaches are based on Deep Learning techniques. The following sections serve the purpose to give profound understanding concerning some key terms of this topic:

- Deep learning
- CNN, R-CNN, Fast R-CNN, Faster R-CNN
- Region Proposal Network (RPN)
- Learning Rate

According to LeCun et.al. [42] **Deep learning** has improved the state-of-the-art in several domains like object detection as well as speech recognition by allowing computational models consisting of multiple layers for processing to learn data representations with multiple abstraction levels. With the help of deep learning complex structures can be found in huge data-sets by using the backpropagation algorithm. While conventional machine-learning methods rely on the processing of data in its raw form, more and more applications of this type make use of deep learning techniques. By applying representation-learning techniques, raw data is used to automatically detect representations needed for further processing (detection, classification). Deep learning fits into this category because the raw input is transformed into multiple representation levels which allows very complex functions to be learned.

CNNs (Convolutional Neural Networks) are heavily used for image classification by reducing dimensionality without a big loss of features and are essential in the context of the next described approaches[9]. **Faster R-CNN** is a methodology also used for real-time object detection because of its high performance. The predecessors of the Faster R-CNN algorithm are R-CNN and Fast R-CNN. In the article of Gandhi [8] it is described that these algorithm are necessary for object detection since standard convolutional networks along with a fully connected layer are not able to handle the detection of multiple objects of interest in one picture. A possibility to overcome this problem is to split up the image into several regions and to apply CNN on each of them. This is not practicable since the size and location of the objects can vary and therefore a huge amount of regions have to be selected to cover all potential areas which would be very computationally intense and therefore very

inefficient. R-CNN as well as the Fast R-CNN method approach the above described problem by using selective search to limit the number of extracted regions called region proposals to a certain number.

The **R-CNN** method extracts 2000 region proposals from the input image with the help of the selective search algorithm. Afterwards the proposals are warped into a square and fed to a CNN (convolutional neural network) / convolutional feature map for feature extraction. The resulting dense output layer is consisting of the features that were extracted from the image. Afterwards these features are fed into a Support Vector Machine (SVM) for object presence prediction as well as for the specification of the objects boundaries. This described approach is still computationally intense and no real-time application is possible because of the high number of region proposals per image that are fed to the CNN. Further more the selective search algorithm is not able to improve its performance since it is a fixed method. Therefore no improvement in candidate region proposal detection is possible.

In contrast to R-CNN, the **Fast R-CNN** feeds the input image immediately to a CNN to generate the convolutional feature map. Afterwards the identification of the region proposals is performed by using the selective search algorithm. With the help of a RoI (Region of Interest) pooling layer the region proposals get reshaped into a fixed size for them to be fed into a fully connected layer. Now a softmax layer (softmax function is an aviation function that turns numbers into probabilities [14]) is used for object predictions and boundary box specification. The Fast R-CNN performs the convolution operation only one per input image to create a feature map from it whereas the R-CNN feeds every region proposal to the CNN. Even though the Fast R-CNN method performs far more efficient than the R-CNN, the circumstance that the identification of the region proposals is still realized by using selective search slows down the execution by a significant degree.

Therefore the **Faster R-CNN** targets on eliminating the performance bottleneck the usage of selective search causes. The first steps of feeding the input image to a CNN and generating a convolutional feature map are identical for both approaches (Fast R-CNN and Faster R-CNN). To evade selective search a separate network (**Region Proposal Network** or RPN) is used for region proposal prediction which are again reshaped by the RoI pooling layer before the proposals are classified and bounding boxes are predicted[8]. Since a learning network instead of a time

consuming selective search algorithm is used for region proposal detection the Faster R-CNN can be used be used for time-critical scenarios.

According to Ren et. al. [51] Region Proposal algorithms form the basis of object detection networks for hypothesizing object locations as well delivering objectness scores. Pictures of any size serve input for a RPN that outputs a set of object boundaries that present object proposals along with their score. Region Proposal Networks are fully-convolutional and are trained end-to end. Region boxes also called ”anchors” are ranked and those most likely to contain the wanted object class are proposed.

In the context of training neural network the term training rate is from high relevance. This rate is a so called hyper parameter that controls the adjustment ratio of the weights of the trained networks taking into account the loss gradient. This means that this value controls the speed in which a model is learning. Choosing an inappropriate learning rate can have different consequences. If the learning rate is too low, it takes a long time until convergence. On the other hand when selecting a too high learning rate the resulting network weights may be sub-optimal while the training process was very fast compared to a low rate. If the learning rate is estimated perfectly the model learns fast to best approximate the function given the available resources [3, 15].

2.2 State of the Art and Trends

There are many approaches for Information Extraction from PDF documents but it is still challenging to find the perfectly suited tool for specific cases. Several studies have been performed focusing on different aspects of the topic ”Table Extraction”. The following paragraphs discuss current strategies to approach this subject area. Current literature concerning the data extraction from native as well as scanned PDFs was reviewed and is presented in the following subchapters.

2.2.1 Native PDF Table Extraction

The extraction of content from native PDF files greatly differs from the extraction of imaged typed files since many techniques can be applied to native PDFs that cannot be used when dealing with scanned PDFs. Even though the PDF standard does support tagging to add structural information, hardly any PDF file contains this information. Therefore, relying on this information is no sufficient to successfully extract tabular content from PDF documents.

The Table Organization (TAO) system allows to automatically detect, extract and organize table-information from PDF documents [48]. The systems approach makes use of the k-nearest neighbor method as well as using layout information to identify tables in PDF documents and to generate the extraction result. It relies on the structure of tables, discovers data as well as meta data and is flexible since it is independent of fixed patterns or layouts. TAO generates an enriched version of the extracted data and aims for portable output that facilitates sharing and further processing of the information. TAO uses PDFMiner, a tool for extraction and analyzing textual data from PDF documents, to convert the document into XML format containing several tags about the document’s characteristics. Afterwards the XML file gets parsed and possible table candidates are detected. The third step analyses these candidates to extract the content of the table cells. The extraction process results in a JSON file containing table information as well as additional information of the PDF document. The tool does not exceed similar approaches concerning performance but is able to overcome some issues related to Table Extraction and is able to handle a variety of different document structures. The combination of machine learning technology and the analysis of layout heuristics is deciding for TAO to identify and extract tabular data. After the PDF to XML

conversion with the help of PDFMiner, additional information like coordinates are included and each page is divided into classes like text, text lines and text boxes. The TAO system performs its task based on the provided output of the external tool PDFMiner and initiates its Table Detection process by searching the text boxes for table candidates with the help of distance calculations and the identification of structural relationships. The extraction process is divided into two steps. After finding the table candidates, the cells are identified by text line comparisons and afterwards the cell value is reconstructed [48].

Table Extraction and Understanding System (TEXUS) is another approach researchers came up with to gather information from tables in documents. It consists of a pipeline of four steps where each step focuses on a specific task and the results of every steps can be analysed separately. Beginning with document converting, a XML file gets generated from the given PDF by the open source PDF wrapping software XPDF. Afterwards, potential table regions are classified to separate the tables from the non-tabular content of the document. During the segmentation process the cells, rows and columns are categorized and header lines are detected. Also, complex table structures are recognized in this step. By performing functional and structural analysis on the gained information, header and data cells are examined to determine the reading order of the table. The outcome does only focus on pure data that is completely separated from its presentation layout [50].

According to Ying Liu et al. the Information Extraction research focuses on the extraction of information from text of digital documents even though the most important information of articles is mostly presented in tabular form. For preparing the digital source of data, the detection of table boundaries plays a key role. Since there is no standard defined for tabular presentation of data, the detection as well as extraction of tabular data is a challenging problem. The approach described in their paper focuses on ”sparse line detection” for identifying the table boundaries and applies CRF (Conditional Random Field) and SVM (Support Vector Machines), two machine learning techniques for further processing. By categorizing the document-lines into sparse and non-sparse lines before initializing the table boundary detection process allows to reduce noise and saves time and effort when proceeding with the subsequent steps. The table boundary detection is also enhanced by keyword detection (”Form”, ”Table), which can also be used in corner cases like separating tables which are presented after another. The research was focusing on scientific documents and directly analysed the documents without conversion into a different format in advance. The process described is split into

four phases: line construction, removal of non-sparse lines, removal of noisy sparse lines, labelling of table lines considering keywords. It is described that PDFs offer enough information for Table Extraction since most tables are text-based. The performance of extracting tables can be significantly improved by prepossessing the data and splitting the line-data into sparse and non-sparse lines[44].

Hassan and Baumgartner [34] describe a Table Extraction method that does not rely on ruling lines and indentations. The first step includes the extraction of text and graphical objects from the PDF file by using the Java library PDFBox. The extracted elements include additional information like coordinates, font attributes and metadata. Also, logical information like the reading order is obtained. Data is further processed to receive all objects on each page with their rectangular bounding boxes. Words are categorized into segments (preferably under-segmentation) and vertical clustering is used to group lines together to achieve a segmentation of paragraphs and table columns. After clustering is completed, line-finding is applied on the entire width of each cluster to complete the preprocessing steps. There are three table recognition approaches applied to the different types of tables.

- **Horizontal and vertical ruling lines** When the same vertical lines are crossing multiple successive horizontal lines, this segment is categorized as a tabular grid. To determine if this grid should be classified as table, a validation step is applied after the classification. Tables detected in this step are excluded from the following Table Detection approaches.
- **Horizontal ruling lines** Pre-processing is applied to lines (dotted lines, touching lines) to unify them and they are removed if they are no ruling lines (separators for headers and footers). Afterwards the candidate columns are sorted based on a weighting that depends on the width, height and number of elements it contains. If columns are intersecting horizontal lines, it is checked if the intersected lines represent a ruling line. If this is the case, it is assumed that the width of this line represents the width of the whole table. Then intersecting objects are added to this bounding box and the bounding box is enlarged if necessary, until no more elements can be added (no intersection). After this, the table validation step is applied to check if the detected structure is representing a valid table.
- **Non-ruled tables** This approach starts similarly to the horizontal ruling lines approach but since no intersecting horizontal line is detected, adjacent candidate columns to the left and to the right are taken into account. At first, columns to the left are added to the possible table grid and the bounding box

is adapted. If this step results in a valid table, the next column is processed until the validation fails or no column to be added is available. Then the same process is applied to the column to the right until the stopping criteria is met. These steps are repeated until the table cannot be expanded any further. If the result is no valid table, all columns, except of the initial one, are returned to the possible table column list. The initial column is dismissed and the whole process is reinitiated with the next column in the possible table column list.

Within the validation step it is checked if the table is valid or likely to be categorized incorrectly. Also some general assumptions are evaluated (same font size, at least two rows and two columns).

Kern et al. [40] describe two approaches for decomposition, both based on unsupervised machine learning techniques. To detect the table regions, contiguous text blocks are extracted from the given PDF files. In the next step, table captions are recognized, and the neighbouring sparse blocks are merged recursively to finally compose a table. Captions are recognized when a line start matches with a predefined list of keywords containing Table, Tab, Tab. followed by a number. Sparse blocks are detected when certain thresholds are matched (width is smaller than 2/3 of the average width or the distance between words is double the average distance between consecutive words. Then the captions are detected, the closest sparse block that is overlapping horizontally with the current caption is selected as the first table element and neighbouring blocks are added to the region if they are matching certain criteria.
To extract the tabular structure two different approaches are compared. The first technique performs hierarchical agglomerate clustering on the words positioned inside the table region to group them into columns and rows to finally achieve a tree structure. The merging process continues until inter-cluster distance reaches a certain predefined threshold. The columns are detected traversing the tree with breadth-first technique and checking if nodes should be split or not until every node has been parsed. For row identification the coordinates of the upper and lower bound of the words are used for 2-dimensional clustering. Afterwards, the clusters are split vertically, and the content of the table cells are detected based on their intersection with the calculated columns and rows.
The second technique for tabular structure extraction relies on partitioning based projection histograms. The rectangular bounding boxes of all words inside the table region are used to calculate vertical and horizontal projection histograms. The boundaries between columns and rows are displayed as minima in these his-

tograms but also some false positives can occur, since not all minima represent cell boundaries. Therefore, these incorrectly classified minima are eliminated in three steps. At first, a filter is applied, that allows to get rid of minima which occur because of single spaces between words. In the next step, the extrema are extracted and non-significant ones are removed and the differences of extrema and their neighbouring extrema are calculated and the list containing the extrema is adjusted. Those clusters that represent tabular boundaries are chosen and the k-means algorithm is applied to the histograms and the table clusters are selected based on the results of the k-means computation.
The described detection approach is generally able to detect the complete tables if they were labelled correctly but fails when the table caption is missing since the Table Detection approach depends on the presence of captions. It occurs that additional blocks of text are added to tables if they are classified as sparse blocks or span across the border of the table. Concerning the structure recognition of the tables, the projection histogram method outperforms the method based on word clustering. This shows that space information is more reliable for table structure recognition than the analysis of overlaps between words[40].

2.2.2 Scanned PDF Table Extraction

Since scanned PDFs consist of images and no real textual data is contained [46], approaches dealing with this type of documents need to perform some sort of pre-processing before being able to extract the actual tabular content.

Deivalakshmi describes an algorithm that does not only identify and remove unwanted lines that can occur due to accidental remarks or improper scanning, but also deals with table segmentation and extraction. Before the tables can be detected and extracted, the scanned documents are pre-processed. This includes the removal of noise, contour smoothing and binarization. To identify the tables of the document, a single mask-based algorithm is applied, only detecting the top left and bottom right corners. Due to this, the algorithm is independent from different line thicknesses as well as language. Additionally, the pre-processed image is transformed into a Pseudo Diagonal Image (PDI). The PDI is now rotated and a line removal algorithm is applied to get rid of unwanted lines of the document image. The downside of this approach is that not only the unwanted but also the meaningful lines are removed. In order to overcome this issue, the already extracted

tables are reintegrated in the ”line-free” document. PDI is used to prevent data loss during the rotation process. A drawback of this algorithm is the fact that e.g. underlines in the actual document image are removed, if they are not part of a table. The good computational performance occurs due to the application of a single mask-based algorithm for Table Detection [27].

Burcu Yildiz et al. describe Table Extraction as a non-trivial topic, which they approached with their tool ”pdf2table”. The work described in their paper indicates that also purely heuristic-based approaches can lead to promising results. ”pdf2table” uses the tool ”pdftohtml” for the extraction of the absolute coordinates of all text elements in the target scanned pdf-file. After the prepossessing step, their task of Table Extraction is narrowed down to semi-structured files with additional information. Based on the attributes returned by the ”pdftohtml” tool, including the coordinates as well as height, width and font of each text chunk, Burcu Yildiz et al applied their table recognition and decomposition heuristics, ignoring graphical components like separating lines. The algorithm is focused on single-column documents and uses following classes for its execution: Text (string), Line (text objects of the same line), Single-Line (line object with only one text object), Multi-Line (line object with more than one text object) and Multi-Line-Block (set of continuous multi-line objects). After sorting the pre-processed data from top to bottom, the text object on the same line are combined by checking if their boundaries are overlapping. The algorithm is based on the assumption that each table must consist of two or more columns. After summarizing lines into multi-line block objects and merging adjacent blocks, the table decomposition step is next. The decomposition step first focuses on column detection by decomposing the multi-line block and proceeds with assigning text chunks to columns. The main limitation of the described approach is the dependence on the results of the external tool ”pdftohtml” since the tool cannot check if the results are incorrect or incomplete. The tool was tested using several pdf documents containing 150 tables and lead to good results for lucid and complex tables concerning the table recognition task. The decomposition part struggled in situations where adjacent cells should be merged because of the need of NLU (natural Language understanding) to reliably perform this task. The approach is domain and language independent, since only structural information is used [53].

Two approaches for automated Table Detection are described by Ranka et al.[49]. The approaches are divided into the recognition of tables with line information and tables with spacing information as boundaries. The assumption on which the

recognition process is based are the following: each table contains a minimum of two rows and two columns, table cells can only contain textual objects (no charts, no tables, no images) and the data provided is binarized and skew corrected.

- **Tables with line information**
  The detection of tables is based on layout, bounding lines and row/column separators, while the reconstruction process relies on intersection points. By calculating the average character height, noise lines are removed, candidate lines of table structure are received and non-tabular images are eliminated. By horizontal and vertical processing of the document lines, the approximate table structures are obtained. Intersection points of tabular lines are used to eliminate non-table lines and reconstruct the table areas. Also, heuristic rules are included into the reconstruction process. The resulting structure does not contain any textual data. Therefore, it is necessary to perform OCR on the defined table regions within the image document to get access to the actual tabular content.
- **Tables with spacing information**
  Method two uses spacing information for table recognition. After splitting the document into separate lines, a spacing threshold is calculated. When gaps between words exist that exceed the threshold, these lines are categorized as tabular lines. The accuracy score of this approach is higher than the score of the first approach but has difficulties when extracting data from multi-column documents.

Huynh-Van et al. [35] propose a hybrid method to identify table zones in document images (for example scanned PDFs). The process is divided into the main parts: region classification, detection of tables with intersecting lines and detection of tables with either horizontal or vertical lines. As a result, this approach is only applicable for the detection of ruling line tables. Based on the region classification, features are extracted that are later used within the machine learning process. Connected components are categorized depending on their size in respect to the image size. Regions are considered as table region candidates if they are part of the convex hull of a large connected component and contain intersecting horizontal and vertical lines. The regions which remain are used for text line identification. Two different sets of features are determined from the previously categorized regions. They improved their model by using Random Forest and Support Vector Machine

based on Scikit-learn[6].

TableBank [43] is a Table Detection approach for image-based document using deep learning and neural networks for its purpose. Li et al. describe the Table Detection and recognition task as a difficult problem due to the high variety of formats and layouts in which tables can be structured. The approach used by TableBank makes use of the Faster R-CNN model that is heavily used in the computer vision area. The Region Proposal Network (RPN) and the Fast R-CNN is merged into a single network by the Faster R-CNN to allow the training of the network in an end to end way. The table structure recognition is achieved with an image-to-text model that includes an encoder for the image input and a decoder of the text output. In order to empower the field of Table Detection and extraction the TableBank data-set is published containing image-based tables in Word and Latex document format [43].

Gilani et al. [31] describe Table Detection as a tricky problem, which most techniques based on layout analysis and hand engineered features fail at, because of the greatly varying layouts of tabular structures. The method Gilani et al. describe in their paper is based on deep learning. After pre-processing the document images, the images function as input for a Region Proposal Network (RPN) that finally results in a connected neural network that is used for detecting tables in the given documents. The algorithm starts with the image transformation step. Hereby the document images are transformed into natural images to be able to use the Faster R-CNN model afterwards. This is achieved using three different methods of distance transformation (distance transformation is used to calculate distances between text regions and white spaces and presents a derived representation of digital images): euclidean distance transformation, linear distance transformation and max distance transformation. For the actual detection step, Faster R-CNN, an algorithm that is heavily used for detecting objects and classifying the in natural images, is used. This process is split in two steps. At first, a Region Proposal Network proposes table region candidates in the form of rectangular objects each having a score. Second the results of the previous step are passed to the region-based object detection module identifying the actual tables and returning their bounding boxes [31].

[6]https://scikit-learn.org/stable/

2.2.3 Overview of State of the Art Table Detection and Extraction Tools

The following Table 2.2 summarized the previously described tools. The name of the tool (if available), the publication year as well as the type of the algorithm is listed and the most significant characteristics are mentioned in form of keywords to give a general overview.

Name of Algorithm	Publication Date	native	scanned	trained	heuristic
pdf2able [53]	2004		X		X
Hassan and Baumgartner [34]	2007	X			X
Ying Liu et al. [44]	2008	X		X	
Kern et al. [40]	2014	X		X	
TAO [48]	2016	X			X
Deivalakshmi [27]	2017		X		X
Ranka et al. [49]	2017		X		X
Gilani et al. [31]	2017		X	X	
TEXUS [50]	2018	X			X
Huynh-Van et al. [35]	2018		X	X	
TableBank [43]	2019		X	X	

Table 2.2: Overview of the in Sections 2.2.1 and 2.2.2 described Table Detection and Table Extraction algorithms. Their area of application (scanned or native PDFs) as well as their type (trained or heuristic) is described.

In Table 2.2 it can be seen that there is a big variety of approaches for the topic of Table Detection, each approach coming along with advantages and disadvantages depending on its characteristics. For native as well as scanned PDFs, heuristic as well as trained Table Extraction techniques are used. The algorithms reach from layout dependant (analysis of coordinates, clustering, sparse line/block detection etc.) over to machine learning approaches based neural networks, region proposal networks and the Faster R-CNN model.

3 Method

The purpose of this chapter is to give a deep insight into the development process of Table Extraction tools. Two different approaches for tabular structure recognition are implemented to be able to handle scanned as well as native PDF documents. For this reason two completely different methodologies are used to exploit strengths and avoid weaknesses of the provided PDF input type. While native documents are processed by a heuristic approach, the Table Detection algorithm used for scanned PDFs is based on deep learning. Both techniques are described into detail, its results are collected and analyzed and advantages as well as disadvantages are discussed. Furthermore, to compare the performance an evaluation data-set consisting of scanned as well as native PDF documents is put together. Afterwards various testing iterations are conducted and based on the results several statistics are generated.

The method chapter is split into two main parts:

1. development of a tool to detect and extract tables from native PDF documents
2. development of a tool to detect and extract tables from scanned PDF documents

For both development processes following aspects are discussed:

- used technologies
- Table Extraction process starting with receiving a PDF file until the export of its tabular data to a .csv file
- algorithms used to preprocess input files, detect tables in preprocessed data and reassemble the data into tabular structure before performing the export
- advantages and disadvantages of both approaches / scenarios
- issues / challenges when developing a Table Extraction tool
- differences that have to be considered when handling native or scanned PDFs

The following Figure 3.1 shows the sequence of steps that have to be taken to from receiving the input document to delivering the structured output in form of .csv files. The actual actions taken in each phase are differing a lot between the two approaches that will be described in the following chapters. Nevertheless from high-level perspective the sequence order is the same for both approaches.

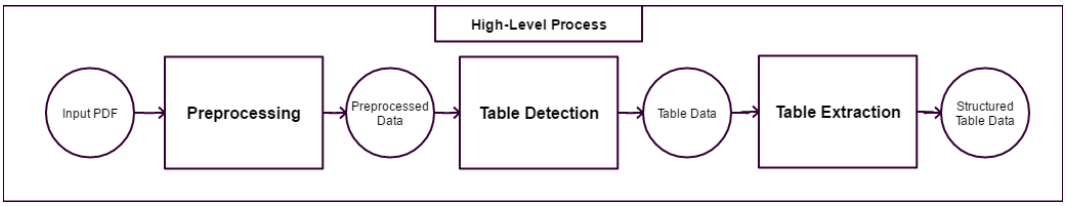


Figure 3.1: This Figure shows the process steps that are completed to generate structured tabular data given a PDF document as input. Even though the actual implementation of the processing steps are very different, from high level perspective both approaches can be split into the Preprocessing, Table Detection and Table Extraction phases.

The following chapters will give deep insights in the implementation of each processing step mentioned in Figure 3.1 for both described approaches. Although the high level overview can be summarized for the heuristic as well as the deep learning based approach, the actual implementation is completely different starting with different types of input documents (scanned and native).

3.1 Native PDF Table Extraction

This chapter describes a heuristic approach for Table Extraction from native PDF documents. The used technologies are listed and a detailed explanation of the implemented algorithms is given. The whole process of PDF Extraction, Preprocessing, Table Detection, Header and Footer Detection and Table Extraction is described in detail. Furthermore challenges coming along with this tasks are pointed out and several Figures are used to help understand certain scenarios.

3.1.1 Technologies

For the implementation of the Java based table recognition tool the IntelliJ IDEA IDE from JetBrains [1] is used. Java was selected as programming language because of Apache PDFbox, an open source Java library that is a very popular and widely used in the PDF extraction and manipulation field.
The high popularity among developers is based on its wide range of functionalities. The library allows the creation of PDFs, the manipulation of existing PDF documents and the extraction of content from PDF documents. Also filling and extracting data from PDF forms and digital signing is supported by the Apache PDFBox library. A key component when developing a native PDF Table Extraction tool is the text extraction engine that allows the extraction of textual elements and several attributes like their coordinates, height, width, Unicode and much more [2][3]. Since the actual detection and extraction algorithms are implemented using standard Java version, no other library apart from PDFBox is necessary for the development of the heuristic Table Extraction tool.

3.1.2 Implementation

In this section the algorithms implemented for every process-step (Preprocessing, Table Detection, Header and Footer Detection and Table Extraction) are explained in detail. Also the method of implementation is described and arising problems and alternative approaches are discussed. The presented approach of Table Extraction from native PDF documents is split into three sub-processes which use the results of the preceding step for their further processing. The PDF Preprocessing step uses the given native PDF document as input and hands over the preprocessed Meta-Data to the next step: Table Detection. Given the Meta-Data, the Table Detection step performs operations to hand over the necessary table data alongside header and footer information that is needed for the Table Extraction phase to reconstruct and export the tabular data in form of .csv files.

[1] https://www.jetbrains.com/idea/
[2] https://pdfbox.apache.org/index.html
[3] http://www.pdfbox.org/pdfbox-user-guide/text-extraction/

PDF Preprocessing

The objective of the PDF Prepossessing step is the preparation of input data in form of native PDF documents for the following Table Detection process. Hereby Meta-Data containing basic but crucial information like the words and lines along with their coordinates, boundaries and other attributes is prepared. Figure 3.2 illustrates the steps that are taken to collect the necessary information.

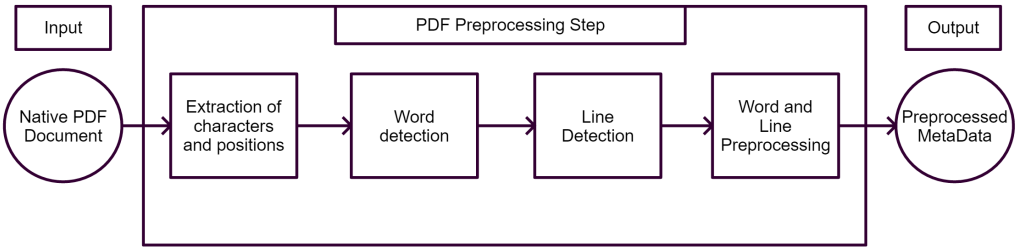


Figure 3.2: The process steps of the preprocessing phase are illustrated. With the help of the external library PDFBox the characters and its positions are extracted from a given native PDF document. Afterwards word and line objects are build by analyzing character coordinates and boundaries to result in enough Meta-Data information to proceed with the Table Detection process after the preprocessing is finished.

As outlined above, the goal of the Preprocessing stage is to gather data concerning words and lines of the given native PDF document. Since the PDFBox library does not provide the functionality to extract text by word or line from native PDFs, the extracted characters have to be analyzed and processed manually to achieve this. Therefore after extracting text elements (single characters) from the file, their position on the page as well as their width and height are calculated based on their X and Y coordinates which are passed on by the extraction process of PDFBox. Next, the letters are combined to words using the spacing Tolerance attribute offered by PDFBox. The spacing Tolerance attribute represents the minimal space width of the current document. When the distance between a character and its successor is smaller than the spacing tolerance and their horizontal boundaries overlap, these characters are summarized to a word.

Figure 3.3 displays the bounding boxes, demonstrated by red rectangles, which are key elements for the word summarization step. Furthermore the maximum values (top, bottom, left, right) of the bounding boxes of all chars of a word are

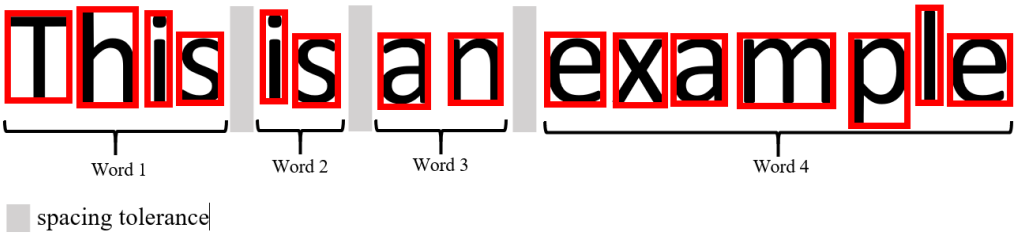


Figure 3.3: In this Figure the summarization of words given the coordinates from the PDFBox export is visualized. The spacing tolerance value is used to identify white-space areas between characters that are likely to separate two words. If the white-space region is smaller than the spacing tolerance value, the characters are combined to one word as long as the tolerance distance is not reached or a new line is observed.

used as the bounding box of a word. This is especially helpful when entering the line recognition stage.

After a word was found, its start and end coordinates as well as its upper-bound and lower-bound values are calculated based on all characters composing the word object. Since for the Table Detection step not only words but also the lines composing the document are very relevant, the next step deals with merging words to lines. This is achieved by comparing the upper- and lower-bound of words and merging them together into the same line if they are overlapping as illustrated in Figure 3.4. To provide additional information required for the further Table Extraction process, the last word of a line is marked and each word also stores its distance to its successor and predecessor if available.

A disadvantage coming along with this approach is the high dependence on the external library PDFBox since all performed calculations are based on the results (coordinates of characters) of the PDF extraction process. In case the PDFBox library passes on incorrect information concerning the coordinates of the extracted characters, the algorithm can produce inaccurate results since word and line boundaries can not be calculated precisely.
To simplify the further processing done in the Table Detection phase, every object (word, line) contains an attribute pointing to the predecessor and successor of the current object (implementation note: currentWord.lastWord is a reference to the word prior to the current one, currentWord.nextWord is a reference to the following word - the same principle is applied to lines). Like this the implementation of

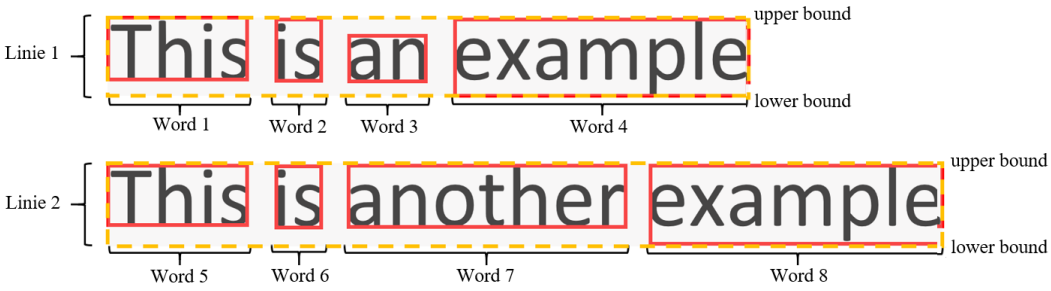


Figure 3.4: It can be seen how line recognition works, given the computed bounding boxes of the words. Hereby the subsequent words are checked if a vertical overlap exists. If this is the case, the words are part of the same line element. As soon as two words are not vertically overlapping a new line is constructed with the initial bounding boxes according to the first word inserted.

clustering and analyzing mechanisms used in the Table Detection step are made easier to implement and the source code stays more readable.
Further more the median width of all lines of a document is calculated for being able to categorize lines based on their structural data in the Table Detection step. As described in the Table Detection process step, the availability of the median line width value allows to recognize lines that are likely to be part of a table. Finally the distances in between lines is calculated which is from high importance when splitting consecutive tables from each other based on the variance of line distances as well as for detecting headers and footers of tables.

Table Detection

The Table Detection stage aims at detecting blocks of lines that combined represent a tabular structure. Given the Meta-Data that is passed on by the Preprocessing stage, operations to categorize sparse and non-sparse lines are performed and the document is categorized single or multi-column. Given the outcome of the sparse line recognition, candidate table areas in form of sparse blocks are computed and subsequently parsed to remove incorrectly categorized lines and blocks of lines. After finishing the sparse line and sparse block evaluation and removal, header and footer detection is performed to pass as much valuable information as possible to the Table Extraction phase.

The Table Detection process is initiated by **detecting the sparse lines** given the

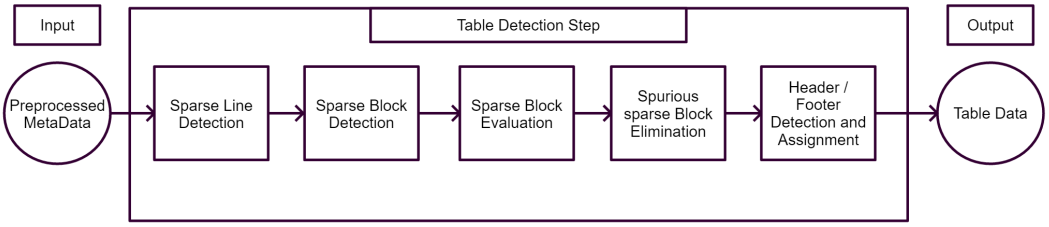


Figure 3.5: This Figure gives a general overview about the process steps of the Table Detection process. Given the preprocessed Meta-data from the Preprocessing phase, sparse lines and sparse blocks are detected and spurious elements are removed. In the end table header and footer elements are detected and assigned to the associated table.

resulting data of the Prepossessing process containing categorized word and line objects having attached basic information like coordinates and boundaries. Sparse lines are labelled as such if the distances between words in the line are higher than a threshold (median space-gap * 1.3). As it can be seen in Figure 3.6, only the distance in between words is considered for sparse line detection purposes whereas the distance of the last word of a line to the end of the line is not. Like this the algorithm does not incorrectly categorize endings of paragraphs and headlines as sparse lines. This saves effort later when incorrectly classified sparse lines are removed. Instead of marking these lines as sparse, they get flagged as ”Critical”. Like this lines of this type which may be endings of paragraphs or headlines but may also be part of a table can get a special treatment as we proceed into the Sparse Block Detection process step.

Additionally, following line types are excluded from sparse line categorization because of their irrelevance for Table Detection as well as their conflicting behaviour with the applied sparse field evaluation algorithm that will be described later on.

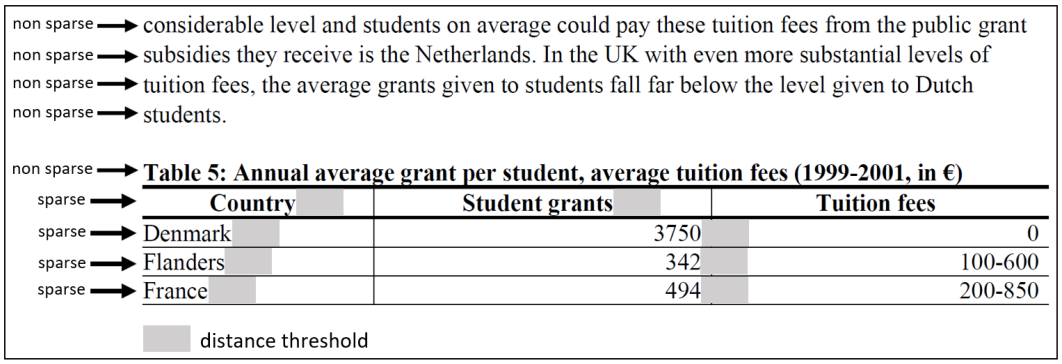


Figure 3.6: To detect sparse lines, the distance between words is compared against the predefined distance threshold. If the distance in between words is as wide or wider than the predefined threshold, this area is categorized as sparse. A line containing sparse areas is a so called sparse line.

- **Bullet Points**
  Lines starting with bullet points are excluded since table columns are not expected to contain bullet points. This is relevant because lines containing bullet points are categorized as sparse lines due to the usually wider white-space element following a bullet point (Figure 3.10).
- **Page Numbering**
  Lines containing the numbering of pages (mostly the first or last line of a page) are excluded since these lines typically do not contain relevant content for the Table Detection task. Not excluding these type of lines often leads to incorrectly categorized sparse elements because of the white-space distance between footer and page number as well as their mostly intended character.

Lines that are categorized as sparse have a count attached, indicating the amount of sparse areas inside each line. This count helps when eliminating incorrect table rows and when building row and line boundaries in the Table Extraction phase. Word groups are built, containing continuous words in lines which are not separated by a sparse field for a more comprehensible and clean implementation of the following steps. Lines marked as critical are reviewed again after having categorized all sparse lines containing sparse fields. For a line to be critical, the length of the line must be lower than two thirds of the median line width. This flag is included to be able to recognize table lines that do not include any sparse point like it can be observed in Figure 3.7. It is reviewed if a critical line is positioned next to lines containing

sparse fields. If so, these lines are relevant for the sparse block detection phase because they potentially are part of a table that might contain lines without a sparse field.

	Participation	Attainment	EEI	GPI	Overall rank
Netherlands	3	3=	1	1=	**1**
Finland	1	8	5	5=	**2**
UK	5	5=	2	5=	**3**
US	7=	1	7	12	**4**
Canada	7=	2	3=	10=	**5**

Figure 3.7: This Figure demonstrates a situation where the second line of the table (containing ”rank” as table data) would not be categorized as sparse since there is no second element in this line. This would lead to wrong results because the table would be split before the third line. Since the second line is intended, the line is reevaluated after the initial line classification and classified as sparse because the line is situated next to sparse lines.

To overcome the problem pictured in Figure 3.7, the approach described above of looking for neighbouring sparse lines is used for categorization. If there are no sparse lines situated next to a critical line, the critical line is irrelevant for further processing. Otherwise the critical line will be treated afterwards at the Sparse Block Detection stage.
The Sparse Line Detection stage is finished when all sparse lines as well as critical line candidates are analyzed and classified correctly. The gained knowledge about the given PDF document from the Sparse Line Detection phase is then further processed in the Sparse Block Detection step after the PDF document is analyzed regarding its structure.

A big challenge when dealing with Table Detection is the different style in which documents may appear. Many Table Detection tools available are not able to distinguish between single-column and multi-column documents by itself. Tools built to analyze just a specific type of PDF document typically deliver poor results when treating various different types of documents concerning Table Detection because of their differential structural characteristics. Multi-column documents for instance do contain a typically continuous vertical sparse area in the center of the page that leads to wrong categorization of lines like it will be described in the Table Detection process step. While many solutions offer the option to pass the amount of columns

to the tool as additional parameter to simplify the process, the algorithm described in this paper allows the automatic categorization of documents as single-column or multi-column. Even though this process step would thematically fit into the preprocessing phase, this step is performed during the Table Detection phase since the sparse area recognition has to be performed in advance. Like demonstrated in Figures 3.8 and 3.9, the left and right page boundaries are determined after eliminating the outliers that may include page numberings or pictures. Given the left and right page boundaries the horizontal center of the page is calculated. This value is relevant to detect multi-column documents since in seriously structured scientific multi-column articles both columns are from the same width. This allows to determine the position in which the field of separation has to be situated in case a multi-column document is analyzed. Based on the computed center value all lines are parsed and marked as ”center-sparse” if a sparse field is situated exactly in the horizontal center of the page. This is done by looping over all word-groups of every line while observing if the end or start values of each word-group full-fill certain criteria (positioned in a certain range from the center coordinated). After all lines of a page are categorized as ”center-sparse” or not, the percentage of lines of the page that are ”center-sparse” is calculated. If a certain threshold-percentage is reached (in this thesis 60% of all lines of a page are ”center-sparse”), the page is marked as multi-column and the next processing step can be initiated based on the gained knowledge regarding the structural type of the document.

After the sparse lines are correctly classified, suspicious text areas are dealt with and the document is categorized regarding its column-structure, the **Sparse Block Detection** stage is reached. This stage aims at identifying continuous blocks of sparse lines which are likely to represent a table in the original document. In order to accomplish this, several factors additionally to the continuity of the lines have to be considered.

Sparse blocks need to consist of at least two sparse line elements since tables consist of at least two rows. Generally these two initial lines are successive sparse lines. It can also occur that the initial lines are separated by a non sparse line in certain scenarios. This case is considered during the sparse block detection stage with the help of the already mentioned ”critical” flag certain lines have attached.

In many cases sparse lines appear randomly after one another in documents without actually being part of a table. With the help of techniques described in the Sparse Line Detection process (handling of Bullet Points, Page Numbering and Critical

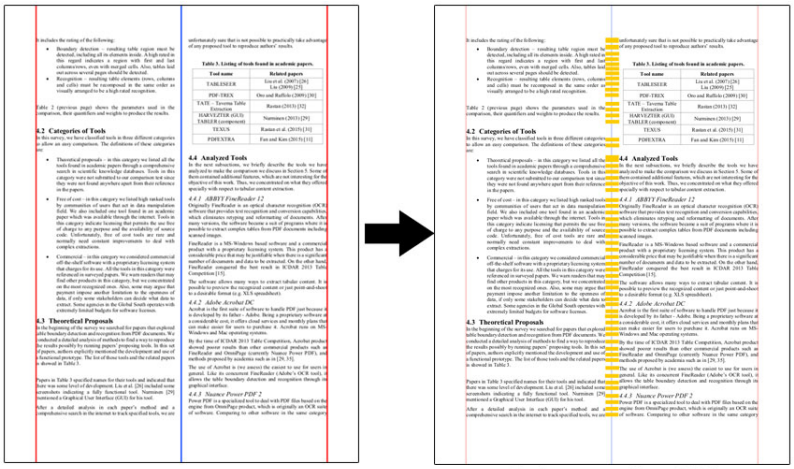


Figure 3.8: In this figure the process of multi-column detection is pictured. The red lines are representing the text boundaries based on which the center value (blue line) is calculated. Afterwards all lines get analyzed if they contain a sparse point in the center area of the page (yellow rectangle). In this example this holds true for 90 percent of the lines. This leads to the page being categorized as multi-column.

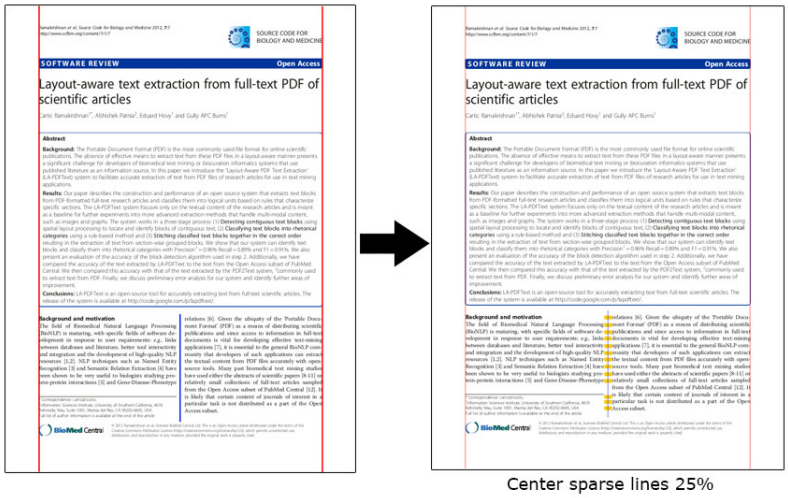


Figure 3.9: In this example only 25 percent of all lines hold true for the condition of being center sparse because of the majority of the page being single column. Therefor this page is categorized as single column since the threshold of 60 percent is not reached.

Lines), the number of incorrectly classified sparse lines is already minimized. Though, in certain scenarios the described techniques are not enough to sort out spurious sparse lines. This happens for example when justified text is analyzed (Figure 3.10) which leads to the frequent appearance of wider white spaces between words because line content is stretched to fit the width of the page.

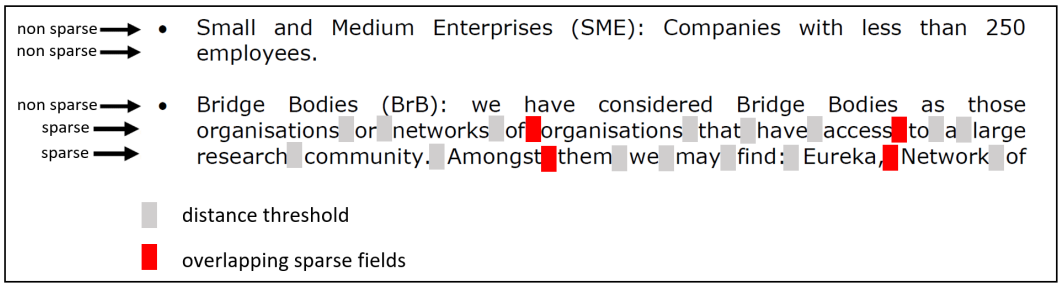


Figure 3.10: When justified text sections are analyzed, often sparse fields are detected that are not part of a tabular structure. This image shows how such scenarios are handled by looking for horizontally overlapping sparse fields. It is also shown that lines being started with a bullet point character are not considered when performing sparse line categorization.

For this reason the spurious sparse lines have to be handled in the Sparse Block Detection stage. Therefore the sparse fields of successive sparse lines are compared to evaluate their adequacy. If sparse fields are overlapping, or in other words, if the white-space areas of both lines in between words or word groups are covering the same areas, the sparse block is considered valid if another criteria also matches. As demonstrated in Figure 3.10 by the red rectangles, it may also happen that randomly sparse fields overlap even though no table is present. Those scenarios represent a big challenge since there is no straight forward solution. To overcome this, a threshold is defined that sets a minimum percentage of overlapping sparse fields in two consecutive lines. In the example of Figure 3.10 the overlapping sparse field percentage is about 10 percent. This results in the sparse block being discarded even though the block is consisting of consecutive sparse lines. In case that there are no overlapping sparse fields at all, these lines are also discarded for the Sparse Block Detection stage and the process continues with the next document line. Generally speaking a sparse block needs to consist of at least two sparse lines that do not need to be consecutive since they could be separated by one or more critical lines. If the sparse block only consists of two or three lines, further analysis concerning its

structure is conducted. Thereby the amount of sparse fields is compared to minimize the risk of incorrectly categorized table blocks.

Following listing gives an overview about the criteria that has to be fulfilled by sparse and critical lines to be part of a sparse block:

A sparse line hast to fulfill one of the following criteria to be part of a sparse block:

- line is not first line of page (except if the next line is sparse or critical and has the same line height)
- line is not last line of page (except if the previous line is sparse or critical and has the same line height)

A critical line hast to fulfill one of the following criteria to be part of a sparse block:

- previous line is sparse (except if the critical line is the first line of the page)
- there are only critical lines between the current critical line and the previous sparse line (except if the current critical line is the first line of the page)
- next line is sparse and line height of current critical line is the same as the line height of the next line
- there are only critical lines between the current critical line and the next sparse line (except if the current critical line is the last line of the page)

After the detection of sparse blocks is completed, the majority of steps concerning the Table Detection phase are completed. Before continuing with the Header and Footer Detection, the found sparse blocks are parsed and evaluated once more in the **Sparse Block Evaluation and Spurious Sparse Block Elimination** stages to get rid of sparse blocks which are incorrectly categorized. Tables which are not fulfilling the required criteria are split, shortened or deleted. This especially affects sparse blocks which are just composed of critical sparse lines that do not contain real sparse fields. This case often occurs when handling papers that include additional administrative information (address, postal code, etc.) that is formatted in a way that meets the described criteria.

Following steps are taken in the Sparse Block Evaluation and Spurious Sparse Block Elimination phase if they are necessary:

- sparse blocks get trimmed
  The line height of the lines at the beginning and at the end of each sparse block are compared to the median line height of the analyzed sparse block and if the difference in line height exceeds a certain predefined threshold the sparse block gets shortened by the affected lines.
- sparse blocks get split
  It is analyzed if the sparse block contains two lines that are separated by a significantly higher distance than the median line distance. In this case two new sparse block are build and the original gets removed.
- sparse blocks get reevaluated
  Since the trimming and splitting of the existing sparse blocks in the preceding steps may result in new results, these have to be reevaluated. Hereby sparse blocks get checked against the minimum length as well as the other criteria described in the Sparse Block Detection stage.

Finally when sparse blocks are identified, trimmed, split, reevaluated and spurious sparse blocks are eliminated, the **Header and Footer Detection** process is initiated. Since this process is started after the table areas are already identified, specific techniques can be used for the identification of headers and footers that could not be used if the process sequence would be performed the other way round. In that case header and footer recognition can hardly be implemented based on the analysis of structural attributes of textual elements (e.g. distances in between lines, height of lines), since such scenarios can appear on many positions in a document and it is hard to distinguish between elements describing tables and other elements like normal chapter headlines or the description of figures. To detect Headers describing the content of a table, following techniques are used:

- Keyword detection
  Keyword detection is the most obvious technique and is able to handle the majority of cases very well. Hereby lines above tables are parsed until a certain threshold is reached (for example a maximum of 5 lines above a table). If a parsed line starts with a keyword like ”Table” or ”Figure” all lines between the top of the table and the currently analyzed line holding the keyword and also the current line are categorized as header of the table (this can be seen in Figure 3.11). Most documents, especially scientific research papers which follow a strict labelling concept are covered by performing header detection based on keywords. In the rare case that a table header can not be recognized

that way, other approaches are used for their identification.

- Analysis of distances between lines
  The second step that is initiated if the keyword detection approach is not able to detect the header of a table is based on the textual structure. This approach compares distances between a line and its preceding one. If the distance between a line above a table that lies within a certain threshold and its preceding line is bigger than the average line distance, all lines between the table and the currently analyzed line including the analyzed line are categorized as header lines for the specific table. This method is very useful in documents that include unlabeled tables which can not be recognized based on keywords. There are certain situations where this practice does deliver incorrect results. This happens if no header for a table is specified in the document and there is a headline or something similar randomly situated inside the defined threshold. To deal with this situation, the line heights of all lines that would be included into the header of a table are compared. If the line heights are varying, it is obvious that the compared lines do not belong to the same textual passage (in this case the header of a table). Because of this the lines can be removed from the header field of the table and the table remains without a header.

When implementing footer detection various characteristics of footers have to be considered that differ from the characteristics of headers. When keyword based footer detection is implemented, it has to be considered, that the start of the footer is always situated directly after the table structure. This means that even though a footer is recognized on the basis of a keyword, the end of the footer has to be detected by performing structural analysis as it can be seen in Figure 3.12.

Therefore the footer detection stage heavily relies on the distance analysis in between lines, more than the header detection phase. While, during the header detection phase the line distances analysis is just used if the keyword detection process does not deliver results, the footer detection stage always relies on it heavily. Hereby the distances in between the lines after the table structure and their succeeding lines are analyzed and if the distance is bigger than the average line distance, all lines between the table and the analyzed line are marked as footer of the particular table. Furthermore, to identify the end of a footer area more effectively, the line height is also used as parameter of comparison. If the line height of lines is varying, the footer area detection algorithm stops and adds the lines prior to this

point to the footer field of the table.

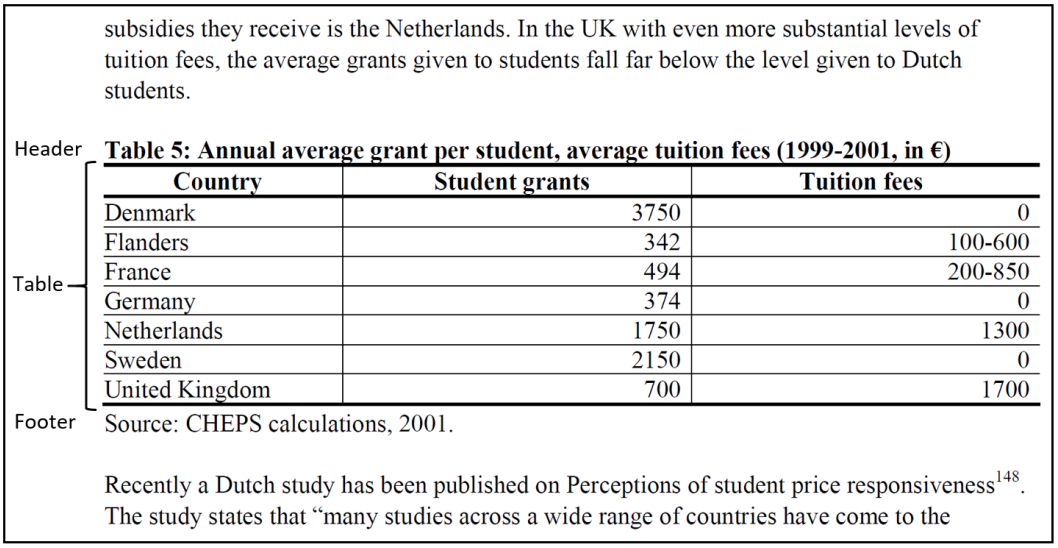


Figure 3.11: In this Figure an optimal scenario for header and footer detection can be found. The header can easily be detected based on keyword analysis while the Footer is clearly separated from the following text passage by a larger than average line distance.

When taking a look at Figure 3.11, it can be seen that the table header as well as the table footer are clearly separated from the actual text before and after the table by a large distance in between lines. Moreover for header recognition the keyword detection approach is successful and therefore the line distance analysis is not necessary. To detect the ending of the footer section, the distance between the first line after the table to the following line is compared to the average distance in between lines of the document. Since the distance is bigger than average a clear footer ending can be determined.

In this process lines with one of the following attributes are excluded from being header of footer candidates:

- lines already being part of another table
- lines already categorized as footer
- lines already categorized as header
- last lines of pages (page numberings)

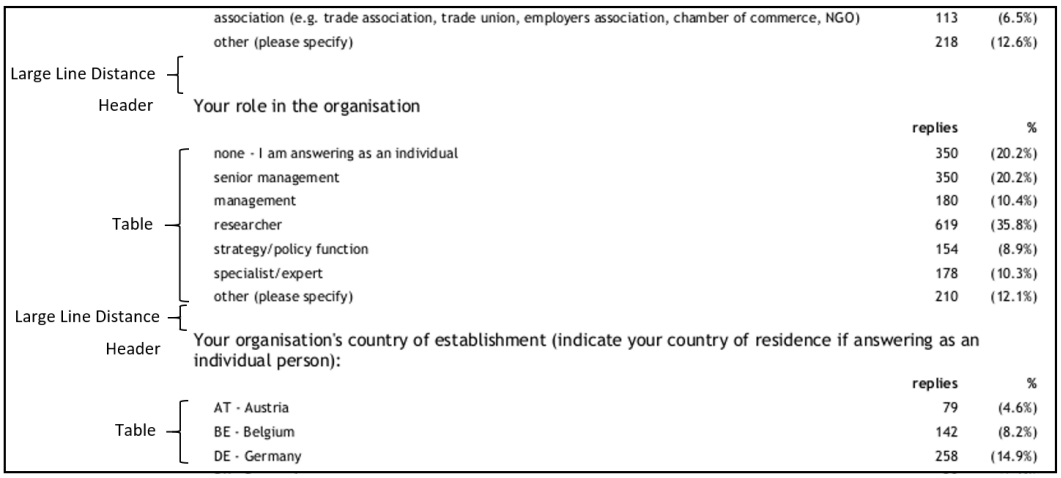


Figure 3.12: When no keywords are available for header and footer detection, line distance analysis is performed as it can be seen in this figure. Hereby the lines preceding a table are analyzed regarding their distances in between them. This analysis is conducted for header as well as footer detection. Footer detection is initiated after the header detection process is completed.

Excluding the mentioned types of lines allows to reduce the amount of lines that can potentially be categorized as table or footer line of a table and allows to increase the efficiency of the process. In rare cases a problem can arise by excluding lines that are already categorized as header or footer of another table. If the PDF document contains two or more consecutive tables that are just separated by header/footer lines (as it can be seen in Figure 3.12), the algorithm may incorrectly classify footers of tables as headers of the succeeding one because of the processing sequence (first all headers of all tables are detected and after this is finished all footers are detected. If the order would be the other way round or tables are parsed one after another, the problem would be the opposite. Header lines of tables could be incorrectly classified as footers of the preceding table.

After completing the Header and Footer Detection phase, following data is available to be proceeded in the Table Extraction process:

- Sparse Blocks
  Sparse blocks are representing the table data in an unstructured way (no rows and columns are defined, the data is just passed line by line as it is extracted from the pdf). The structural data (words groups, sparse areas, size,

distances, coordinates) will be used in the subsequent process to identify the actual tabular structure holding the data that is included in the passed along sparse block.
- Header and Footer information
  The information concerning the header and footer of the tables included in the given document are already extracted in a form that can be used by the Table Extraction phase as it is. Headers and footers are just considered when exporting the data in form of .csv files since they do not need to be further processed.

Given this information the Table Extraction process is initiated with the goal to correctly identify the tabular structure, discarding invalid sparse blocks that do not contain table data and ultimately exporting the processed tables of the given PDF document in form of .csv sheets.

Table Extraction

The algorithm for extracting tables in the Table Extraction phase given sparse blocks is very complex since many different scenarios have to be considered and a variety of checks are applied. In general this phase can be summarized into following steps illustrated in Figure 3.13.

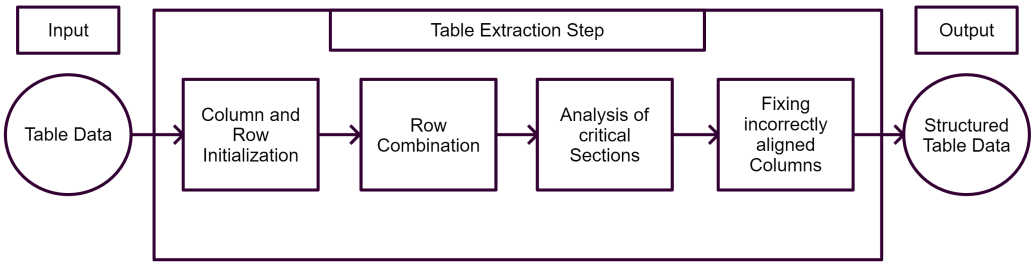


Figure 3.13: The Table Extraction process relies on the unstructured table data that is the result of the Table Detection process. The columns and rows are initialized and combined based on certain parameters. Afterwards critical sections are evaluated and incorrect assignments are fixed. The outcome of this process is structured table data that is forwarded to the Table Export stage.

Figure 3.13 shows the main processing steps that are conducted during the Table Extraction Phase. The illustrated steps are described into detail in the following Sections and Figures are used to give examples of selected scenarios that are treated in the process.

At first the table columns of the given sparse block are **initialized**. The strategy chosen to initialize the columns is very reliable but shows weaknesses in certain scenarios. To identify the initial columns, the given sparse block is parsed and the line containing the highest number of sparse points is taken as indicator. If there are multiple lines having the maximum sparse-field count of the sparse block, the first line is chosen for the initialization of columns. In this step the selected line is taken as input and the columns of the table structure are initialized based on this line. Hereby the start and end coordinates of the words or word groups separated through sparse areas are used as the initialization values for the column boundaries of the treated table. These newly created column boundaries serve as a template for the next processing step in which the sparse block lines are analyzed, split up and their parts are assigned to the correct line and column. Therefore, word groups are compared against the current column boundaries of the table. If the word-group boundaries overlap with the predefined column boundaries, the word group is added to the column and the algorithm proceeds with the next word-group if there is one. After adding text elements to a column, the boundaries of the columns are updated if the start value of the word group is smaller or the end value is larger than the one of the column boundary.

Obviously there might be cases in which a cell might span more than one column. If this is the case, the attribute representing the number of spanned columns by a cell of the element is increased by the number of columns the cell is spanning. Like this when exporting the table-data into an .csv sheet, spanning cells can be correctly represented. If no overlapping column element is found, what is very rare since the row containing the highest sparse count is taken as structural template, the current word group is marked as critical as it is the case in column 3 and column 5 of Figure 3.15. Here all elements of the mentioned columns are marked as critical since the first table line is used as column template.

Additionally, before the column classification process is initiated, the average distance between lines of the block is calculated to perform **Row Combination** afterwards. This targets cells consisting of multiple line text elements since their y-distance often is smaller than the average y-distance of the whole table as shown

in Figure 3.14. By comparing these two values (average y-distance and y-distance between the current and the previous line), textual content that belongs to the same cell can be summarized and extracted in a more convenient manner. In the following Figure 3.14 the distance between the second and the third line is less then the average distance in between lines of this table. This circumstance points out that these lines should actually be combined into one and the content positioned in the same row is combined to one cell.

	Involvement of pupils in		
	Preperation and planing	Production of materials	Presentation and evaluation
Knowledge and awareness of different cultures	0,2885	0,3974	0,3904
Foreign language competence	0,3057	0,4184	0,3899
Social skills and abilities	0,3416	0,3369	0,4303
Acquaintance of special knowledge	0,2569	0,2909	0,3557
Self competence	0,3791	0,3320	0,4617

Figure 3.14: In this Figure the above described case of content belonging to the same cell even though belonging to different lines is demonstrated. During the Table Extraction process the distances in between table-lines are compared and lines being conspicuously close to each other are summarized to one table row (in this example rows two and three).

After all lines of a sparse block are analyzed and classified into columns and rows, the **critical ones are reevaluated**. Hereby the critical word groups are again compared against the current table structure as the structure might have changed and the column now fits into an existing column. If that is not the case (see example Figure 3.15), a new column is inserted between the column left and right of the word group and the whole table structure is updated.

When this process is completed, all textual elements should be part of a table cell that is represented via a row and a column index. Even though this would already lead to considerable results, one more optimization step is executed before the Table Extraction into a .csv file is initiated. This step focuses on **fixing incorrectly aligned columns**. The focus lies on columns that contain just one text element since these are suspicious on being categorized incorrect. All columns are observed and those containing just one data element are selected for further processing. Thereby it is checked whether the horizontally neighbouring cells positioned in the selected lines are empty or not. If they are empty, the cell value of the column

containing one element is transferred to the empty neighbour-cell. Then the now redundant column is deleted and the table structure is updated and ready for its export.

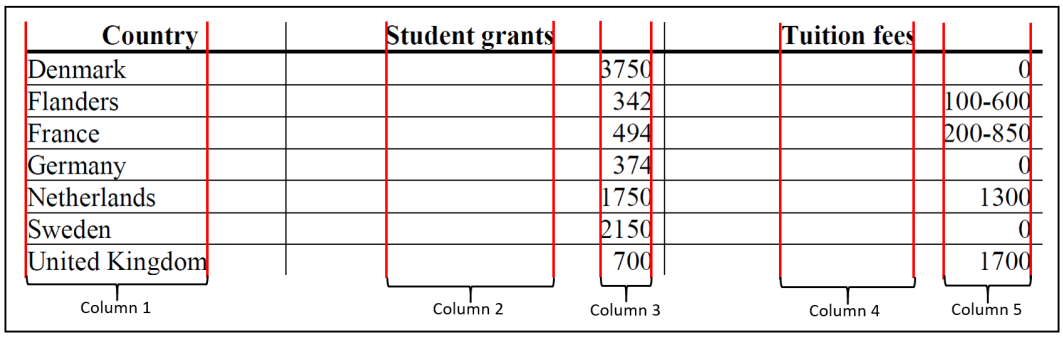


Figure 3.15: In this Figure two challenging situations are displayed. Since all lines have the same sparse count, the first line is chosen as column template. This leads to the circumstance, that all word groups of column 3 and column 5 are marked as critical. Moreover column 2 and column 4 are just containing one element and therefore have to be reevaluated again.

Figure 3.15 demonstrates several critical situations especially concerning column boundary detection. Since the first line of the table is structured in a different way compared to the remaining table lines, on first sight obvious table columns (column 3 and column 5) are marked as critical since they are not overlapping with the column boundary template. Furthermore there are columns containing just one word group which leads to them having to be reanalyzed after the table structure is built. For column 2 and column 4 it is checked if any horizontally neighbouring cell is empty. Since this is the case, columns 2 and 3 as well as columns 4 and 5 are summarized, the structuring process is finished and the table export can be initiated.

Table Export

To perform the export of all tables of the analyzed PDF documents, the free Java library called Apache POI[4] is used. This library targets on offering pure Java ports for reading and writing different file formats. For the purposes of the described

[4]https://poi.apache.org/components/index.html

algorithm a .csv file per PDF is created and filled with the given tabular data. Each table of a PDF file is exported to a separate worksheet within the csv file. Like this the user is offered .csv files containing all extracted tables while each table can be treated separately since they are split up in sheets. Furthermore the detected headers and footers of all tables are exported to help understanding the tables without having to search for the table labels in the actual PDF document. Additionally every sheet containing a table also contains the page number of the table in the original document. Like that the user can immediately jump to the original table in the document if needed. This might be the case for example for suspicious table structures or just special interest.

3.1.3 Problems and Challenges

In this section several problems and challenges that come along with the development of a heuristic Table Extraction tool are described. Certain scenarios are outlined with practical examples and also possible solution approaches are mentioned that are partially applied in the native Table Extraction tool described in chapter 3.1.2. The following listing of problems are categorized depending on the process step in which they have to be solved (Preprocessing, Table Detection, Table Extraction).

- Preprocessing Stage
  - No automated extraction of structured text elements
    When treating native PDFs it is necessary to be aware of its structure. PDF extraction libraries typically allow to extract all characters of a page as well as their coordinates. To structure the extraction results into words and lines an approach similar to the algorithm described in chapter 3.1.2 has to be applied since e.g. the PDFBox library does not provide an implementation of this process step.
  - Depending on an external extraction tool
    Since somehow the input PDF documents have to be extracted to perform further operation based on the extraction results an external tool or library is used. This results in a certain factor of dependence since the results delivered by the external tool are used as bases for all other

processing steps. If the extraction results are faulty (e.g. wrong coordinates of characters are delivered) the whole algorithm is not working properly.

- Table Detection Stage
  - No standardized table format
    The structural composition of tables is from great variance. Because of the broad variety of fields of application and the big variance of types of data that can be represented no blueprint of a tabular structure is existing. This makes it a very complex task to detect tabular structures since no heuristics assumptions will be able to work perfectly for every scenario. Therefore it is necessary to perform heavy testing when determining thresholds and making assumptions that should be able to handle the majority of cases. This regard following points described in the chapter 3.1.2:
    * What is a table
      How many rows and column must a table at least be composed of?
    * Sparse point threshold
      How large is the minimum sparse area in between columns compared to normal text?
    * Line distance threshold
      How much does a line distance have to vary to split sparse blocks?
    * What is a page header/footer/numbering/bullet points
      What do page headings/footer/numberings and other structural elements on pages look like for being able to categorize them appropriately.
  - Separation of consecutive tables
    Separating tables that are positioned without being interrupted is hard since the only indications of a new table starting is the distance in between lines or a variation of line heights. Hereby a threshold is defined that present the minimal variance concerning distance and height for the table to be split. Hereby in rare cases tables that are intentionally structured in a way that a high variance of different distances between lines of line heights appears my be split incorrectly.
  - Exclusion of irrelevant information (page number, heading, etc.)
    To detect header or footer areas of pages rules are applied that should

correctly categorized these lines. If tables are starting in the first line of a page or ending in the last line and are also matching page header or page footer criteria, these lines of a table can get discarded.

- Document orientation
  Depending on the format of the document (landscape or portrait format) the extraction method has to be slightly adjusted.
- Differences in document structure
  Since the structure of documents is strongly varying it is hard to select an algorithm that is able to handle every document at the same level. Documents that include just tables do need to be treated differently to documents consisting of large textual areas. Hereby determining the sparse point threshold that fits both scenarios is the key factor. Also documents that use justified text present a challenge since many text lines are categorized as sparse even though they are not part of a table because the text is stretched to fit the whole line correctly. To minimize the appearance of incorrectly classified sparse blocks techniques described in Chapter 3.1.2 are applied.
- Boundaries of tables are not used as indicator
  Since the described Table Detection approach is optimized to handle native documents the document is just analyzed based in its textual components that can be extracted because of its native nature. This leads to the circumstance that structural elements like table boundaries are not used to perform Table Detection even though they would frequently be a helpful indication of where a table is positioned on a page.
- Multi-column documents with tables overlapping both columns
  The rare case of tables spanning the whole page appearing in multi-column documents requires complex analysis of the page structure.
- Multi-line table cells / multi-column table cells
  Cells spanning multiple table rows or table columns are sometimes hard to detect correctly because the graphical cell boundaries are not used for this purpose. Therefore analysis based on line distances are performed to determine whether a table cell spans one or more columns/rows.
- Header/footer categorization sequence sometimes lead to unexpected results
  It can depending on the situation be hard to differentiate between headers and footers of tables if they are positioned consequently on a document page. Hereby often the processing sequence (at first the headers

and afterwards the footers of all tables are detected. Since lines already categorized as header or footer element (described in Chapter 3.1.2) are excepted from further categorization analysis, already categorized header lines that actually represent footer lines of the proceeding table can get incorrectly categorized.

- Varying distances between lines within tables
  In general varying distances inside a tabular structure indicate either cells spanning multiple lines (smaller distance) or result in the sparse block being split up (bigger distance) like described in Chapter 3.1.2. If there are variations in line distances included on purpose this may lead to a table being split up when a certain line distance threshold is reached even though no splitting would be necessary.
- Charts or images with text
  Charts or images that are equipped with text are hard to differentiate from tabular structures because they generally consist of sparse lines. Therefore false positives may appear that the algorithm is not able to discard.

3.2 Scanned Table Extraction

In this chapter a deep learning based approach for Table Extraction of scanned documents is described. As outlined in chapter 2.1.1, a scanned PDF document does not include any digital code that describes its structure or formatting. These ”image-only” typed documents neither include meta-data information nor is it possible to extract textual content without performing OCR beforehand. The described method takes scanned PDF documents as input and feeds the image of every documents page to a Region Proposal Network after they were preprocessed. Subsequently the Table Detection is performed by a fully connected neural network. The approach described by Gilani et. al. [31] is used as basis for the implementation. According to Gilani et. al. this method is able to handle different typed layout documents like research papers or magazines with high precision and can compete with state of the are Table Detection tools like Tesseract.

3.2.1 Technologies

Since the described approach for Table Detection in scanned PDF documents is based on deep learning, the machine learning system **TensorFlow** was chosen for training of the neural network and prediction. TensorFlow is an open source library developed by Google's Machine Intelligence Research organization with the focus on data-flow graph based numerical computations[17]. Training and inference of deep neural networks are main fields of application for the TensorFlow system. TensorFlow operates in various environments that include GPUs (Graphics Processing Unit), CPUs (Central Processing Unit) as well as custom designed TPUs (Tensor Processing Unit) [18]. Many Google applications and services are using TensorFlow because of the excellent performance TensorFlow provides. The flexibility of TensorFlows architecture allows developers to implement their own optimizations and algorithms for training. Another advantage of TensorFlows architecture is its portability. As already stated it can be executed in multiple environments (GPU, CPU, TPU) but it can also be deployed everywhere if the necessary run-time requirements are met. Adaptions and optimizations can be implemented to provide good performance also on various platforms. TensorFlow uses so called **TFRecords** for the serialization of data. TFRecords is used for storing a sequence of binary records so they can be read linearly [16]. The usage of a binary file format when working with large data-sets results in increased performance concerning the training of the model as well as saving disk space. Further more binary data can be copied and read very efficiently.
Another benefit of using TFRecords is its optimization for working with Tensorflow. Multiple data-sets can be combined and huge data-sets that can not be stored in memory as a whole can be processed by only loading the required data at the time [11]. The computer vision toolkit **Luminoth** that is focused on object detection and built based on TensorFlow is also used for the implementation of the scanned PDF Table Detection method [4]. Furthermore, Luminoth does provide a great graphical interface that shows the results of predictions in a comprehensible way. Because of a huge difference in performance it was decided to execute the training of the deep neural network GPU based. While CPUs are optimized to solve complex tasks with few powerful cores, the advantage of GPUs in the context of deep learning is the large number of simple cores that allow the computation of thousands of simple problems in parallel. Since not the complexity but the number of mathematical operations executed to train the neural network is high, the performance of GPUs

is much higher than when using the CPU cores for processing [2].
The programming language chosen for the development of the scanned Table Detection tool is Python and the used IDE Pycharm offered by JetBrains [7]. Python was chosen since it was the first language that was supported for using the TensorFlow engine and furthermore most features of the framework are currently supported in python even though many features are being implemented to be offered as a C API also. These are the main tools and technologies used for implementing the scanned Table Detection and Extraction tool but a few more will also be mentioned in the following chapters.

3.2.2 Implementation

The implementation of the scanned PDF Table Detection tool is split into several processing steps that will be described in detail in the following chapters. The main task before the actual tool is developed is the training of the Network with the help of several technologies. After the training process is completed a checkpoint is created that will be used for Table Detection in the real program. For detecting the Tables given the scanned PDF files several steps have to be taken to structure the data in a certain way that allows the chosen algorithm Faster R-CNN to perform optimally since this approach is optimized to process natural images [31]. When the preprocessing is finished the actual table boundary prediction is initiated, the results are stored appropriately and the input documents are recomposed after conducting Optical Character Recognition on every document page. After the scanned documents were processed and OCR was conducted the resulting data that is passed to the extraction phase are native documents as well as the table boundary coordinates that were detected. For the Extraction stage native PDFs are necessary because otherwise it would not be possible to perform text extraction. Given the bounding boxes the table areas are extracted and the content is structured to finally being exported as structured data in form of a .csv file.

Training Process

The training process is initiated after the required training data is selected and the files are preprocessed for optimal performance of the Faster R-CNN algorithm. After the preprocessing is finished the data is converted into TFRecords, a TensorFlow

optimized data format. Given the TFRecords the training is started based on the configured parameters. After the Training Loss threshold is reached a checkpoint is manually created that allows to restore the trained model in the following processing phases.

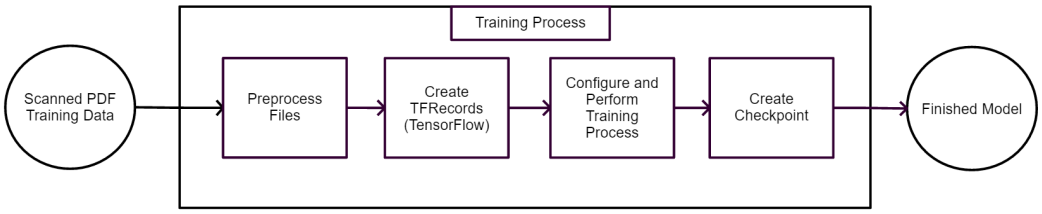


Figure 3.16: This Figure shows the procedure of how the data selected for training purposes is processed. After the data was preprocessed the TFRecords are created before initiating the Training process with the help of the TensorFlows framework. When the training loss threshold is reached a checkpoint is created that stores the current state of the trained model.

Before the training process can be initiated the data used for training purposes has to be chosen. Therefore a data-set consisting of about 500 images along with .csv files containing their ground truth is selected [10]. This image files get categorized into training and validation set randomly at a ratio of 80 percent (training) to 20 percent (validation). While the training files are used to perform the training process of the model, the validation set is used to estimate the performance of the current model. Based on these estimations modifications on parameters like the learning rate are performed to optimize the training. Every image contains one or more tables which are described in the ground truth files like the following:

As is can be seen in Figure 3.17 every line describing a table consists of six elements. The four coordinates (X-Min, Y-Min, X-Max and Y-Max) describe the bounding box of the tabular structure. X-Min and Y-Min together describe the upper left corner of the table element while X-Max and Y-Max are describing the bottom right corner. Furthermore the name of the image in which the element is situated is given along with the label of the object. Since only tables are detected and no other types of objects are relevant for this task, all entries do have the label ”table”. After the data-set along with its ground truth is selected the images have to be preprocessed. To allow optimized processing of the Faster R-CNN method the images are now naturalized. According to Gilani et. al. [31] by applying image transformation the

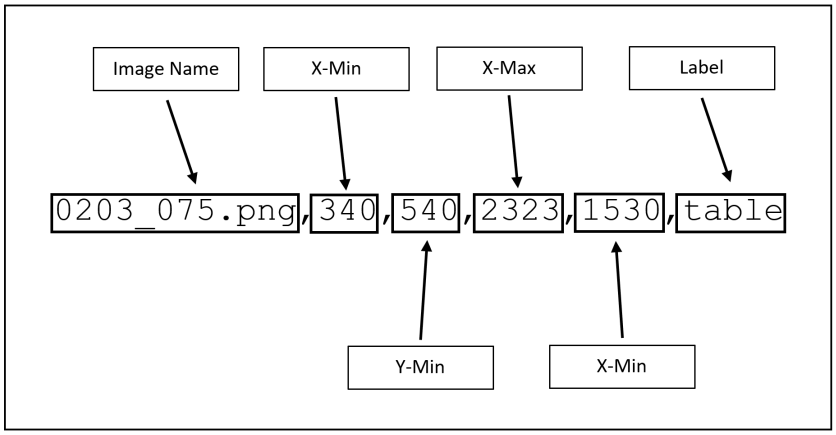


Figure 3.17: The ground truth consists of six elements per table entry. Each table is described by its document name, the boundaries of the tabular structure as well as the label of the object which is in this case always table since only one object type is analyzed.

images can converted into natural images as close as possible. Distance transformation is applied on the images since in the process the distances between text regions and white spaces are calculated which gives good indications about the existence of table regions in the analysed files. As proposed three different image transformation algorithms are applied on the binary input images to store different feature types in the three channels[31]. The applied transformation methods are the following:

- Euclidean Distance Transformation
- Linear Distance Transformation
- Max Distance Transformation

The transformation algorithms are applied by using the OpenCV library, an open source computer vision and machine learning library [1]. After the transformation of the images is finished the three resulting channels (blue, green, red) are merged together (cv2.merge). The following Figure 3.18 shows the result of the applied image transformation:

After all required files are preprocessed the data is binarized by creating TFRecords. These TFRecords will afterwards function as input for the training. For doing so the command line based tool ”lumi” (part of the Luminoth tool) is used. When the conversion of the data into the for TensorFlow processing TFRecords format is finished the training process is configured. Hereby several parameters are essential:

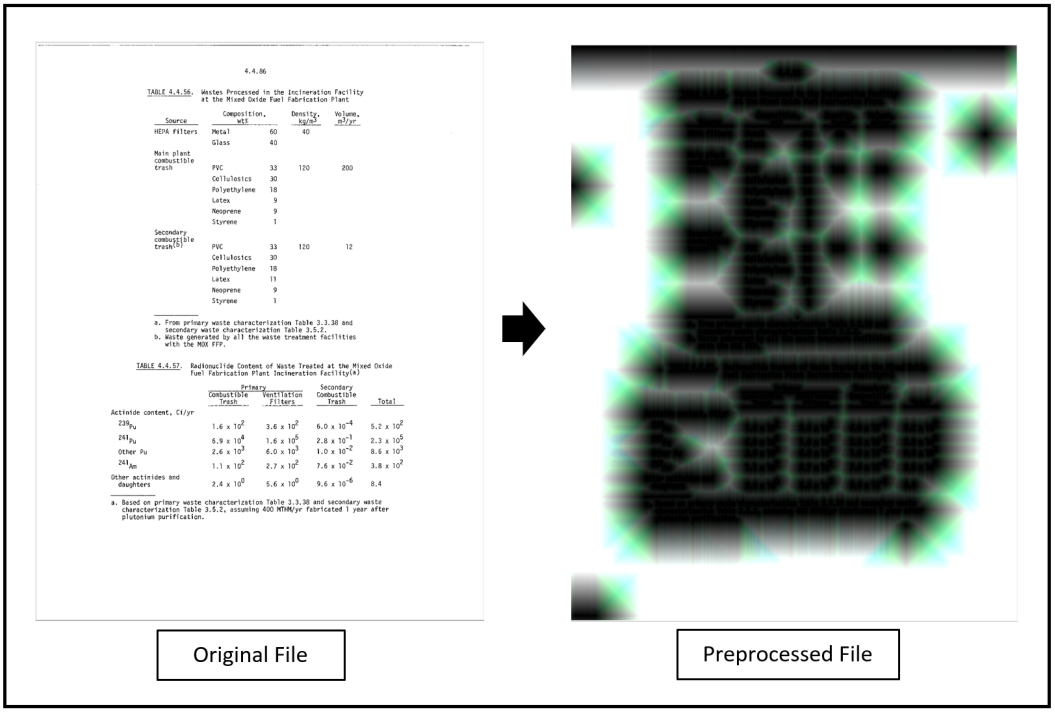


Figure 3.18: After performing three different types of distance transformation on the input image, these three channels are merged together. This Figure shows the result of the described process.

- Number of Classes (one since only table objects are relevant)
- Learning Rate
- Used Model (Faster R-CNN)
- Min and Max Boundaries for Image Size
- Checkpoint saving periods
- Number of Epochs

Based on the described configuration the training is initiated via the command line tool Luminoth. While the training is in process the current training loss values as well as the number of steps already executed are displayed as it can be seen in the following Figure 3.19. Hereby by adapting the learning rate the target is to minimize the training error to receive the optimal results in training the model.

The training of the model is finished as soon as the defined threshold for the training

```
INFO:tensorflow:step: 574, file: b'0101_003.png', train_loss: 3.427340269088745, in 0.73s
INFO:tensorflow:step: 575, file: b'0672_278.png', train_loss: 3.8334827423095703, in 0.78s
INFO:tensorflow:step: 576, file: b'5067_007.png', train_loss: 3.4055306911468506, in 0.73s
INFO:tensorflow:step: 577, file: b'0154_080.png', train_loss: 3.8915839195251465, in 0.76s
INFO:tensorflow:step: 578, file: b'0207_025.png', train_loss: 3.3610756397247314, in 0.76s
INFO:tensorflow:step: 579, file: b'1124_039.png', train_loss: 3.1988673210144043, in 0.81s
INFO:tensorflow:step: 580, file: b'0210_111.png', train_loss: 4.6790618896484375, in 0.74s
INFO:tensorflow:step: 581, file: b'2117_329.png', train_loss: 3.148057222366333, in 0.72s
INFO:tensorflow:step: 582, file: b'1486_156.png', train_loss: 3.861640691757202, in 0.81s
INFO:tensorflow:Saving checkpoints for 584 into jobs/table-area-detection-0.3\model.ckpt.
INFO:tensorflow:step: 583, file: b'1896_414.png', train_loss: 4.947868824005127, in 3.73s
INFO:tensorflow:step: 584, file: b'1201_293.png', train_loss: 3.093141555786133, in 0.74s
INFO:tensorflow:step: 585, file: b'0210_111.png', train_loss: 4.088712215423584, in 0.73s
INFO:tensorflow:step: 586, file: b'1078_082.png', train_loss: 3.0322093963623047, in 1.03s
INFO:tensorflow:step: 587, file: b'1249_072.png', train_loss: 2.9907453060150146, in 0.77s
INFO:tensorflow:step: 588, file: b'0199_384.png', train_loss: 5.524785041809082, in 0.69s
INFO:tensorflow:step: 589, file: b'2070_034.png', train_loss: 4.712167263031006, in 0.75s
```

Figure 3.19: This screenshot shows the displayed information by the Tensorflow based tool Luminoth while executing the training process. Hereby the number of the current step along with the name of the current file is displayed. Moreover the training loss values and the processing time per file are shown.

loss is reached. When this phase is reached a checkpoint is manually created that stores the final version of the model that was created during the training process. This checkpoint allows to always restore the model when evaluation or prediction is performed. This model will be used in the next stages to perform the detection of tabular bounding boxes inside scanned PDF documents.

Table Detection

Once the Training process of the model 3.2.2 is completed the implementation of the actual tool for Table Extraction is initiated. The method is split up into several steps that, given a scanned PDF document as input file, result in a native PDF file and along with a .csv file containing its detected table boundaries. To get to the desired results the input documents are split up and converted into images before they are getting preprocessed. Afterwards the table boundary prediction is conducted and the resulting bounding box coordinates along with the file name and the page number are stored in form of .csv files. Finally the original documents are recomposed as native documents by applying OCR using the Tesseract engine and passed over to the extraction process.

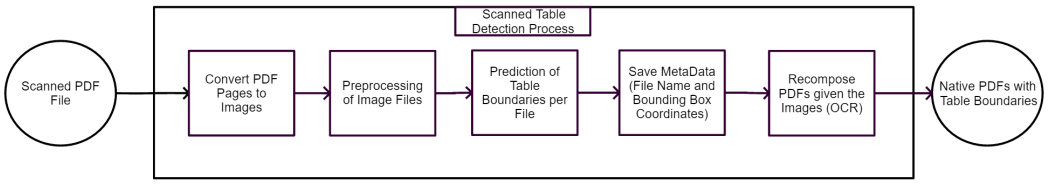


Figure 3.20: The Table Detection process is takes a scanned PDF files as input before converting every file page into an image. Once the Preprocessing is finished the table boundary prediction is executed. The resulting MetaData is stored and passed on to the next stage after recomposing the original documents as native documents by performing OCR.

The program receives **scanned PDF documents** as input files, no matter how many and of what size. After the process is initiated as a first step the input documents are **converted**. Hereby every PDF page is converted into an image because the trained previously described model is optimized for table structure recognition on image files. For that purposes the Python library pdf2image is used [6]. When the conversion process is completed the image files are getting **preprocessed** to prepare them for the table boundary prediction process. As described in 3.2.2 in more detail, the preprocessing is conducted with the help of the open source OpenCV library. Hereby each three transformation algorithms are applied on each image before merging all channels together. Once the transformation and merging procedures are finished the files are ready for the Prediction step. The **Prediction** method takes two input arguments:

- Checkpoint created when the Training process was finished (unique checkpoint ID)
- Preprocessed image file of PDF page (file name)

Given these mentioned parameters the actual table boundary detection is performed. The results of this are stored into .csv files named like the original PDF file in the ”save Meta-Data” processing step. Every entry in a .csv file consists of the following five values:

- Page-number
- Min-X
- Min-Y
- Max-X
- Max-Y

With the help of these attributes the bounding boxes of each detected table can be assigned to the correct document as well as the page it is placed on. The Luminoth tool also offers a simple web application where predictions can be performed using a graphical user interface for visualizing the results. For using the web tool the command ”lumi server web –checkpoint ”Checkpoint-ID” is executed. By passing the checkpoint identifier to the tool the trained model can be rebuilt in the desired state. Once the command is execute a preprocessed document image can be uploaded and the bounding box detection process is executed. The Result of the execution is displayed textually as well as graphically as it can be seen in the following Figure 3.21. The textual output can look like the following:

”objects”:[”bbox”:[600,359,1932,1525],”label”:”table”,”prob”:1.0]

The output parameters of the above shown output line are:

- **objects**
  containing the actual list of table object that were detected
- **bbox**
  the coordinates describing the boundaries of each detected table object
- **label**
  the label of the detected object (in this scenario irrelevant since only one object type is detected)

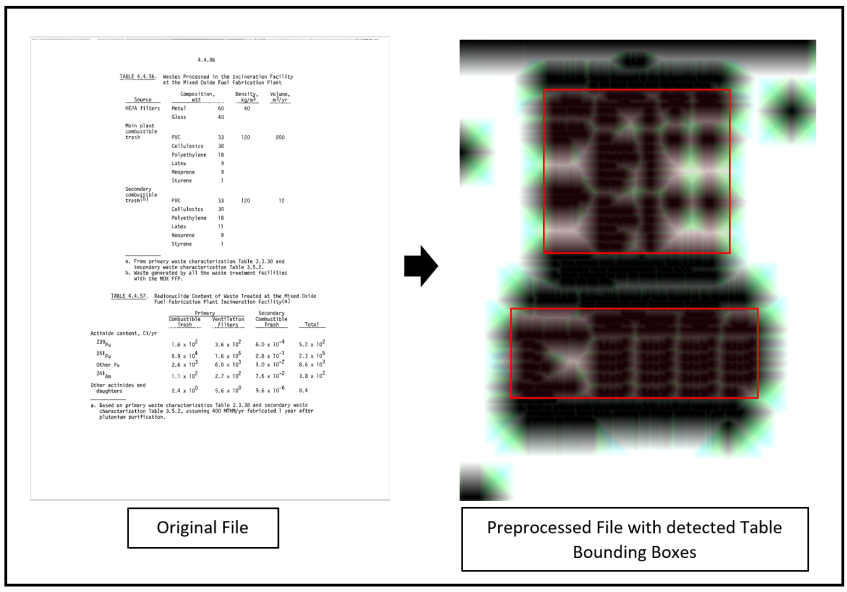


Figure 3.21: This Figure shows an excerpt of the web interface offered by Luminoth. On the left side the original document is shown whereas on the left side the preprocessed file is shown after the table boundary detection was executed.

- **prob**
  represents the probability of a structure to be categorized as an object

Once the prediction is finished and the files containing the boundary data are created and filled with the required information the documents are prepared for being passed over to the Table Extraction stage. Since for extracting textual data from PDF documents the type of the target PDF has to be native, the input documents have to be changed from scanned into native. Therefore Optical Character Recognition is performed for every document page supported by the open source Tesseract OCR engine. After every page was parsed and the textual content is stored the documents are **recomposed** into its original paging structure with the difference of being native typed instead of scanned. Finally the Table Detection stage including the preprocessing as well as postprocessing of the input files is concluded by handing the now native files along with the .csv files containing its meta-data over to the extraction phase.

Table Extraction

The last phase that is put into practice before the structured table data is presented to the user in form of .csv files is the Table Extraction phase. This phase processes the input of the preceding phase (native PDF documents, files containing table boundaries) by first extracting the textual data along with its coordinates. Afterwards the columns and rows are initialized based on the bounding boxes and the rows are combined based on the line distances. Afterwards critical sections are reevaluated and incorrectly aligned columns are corrected before the final table data is exported into .csv files.

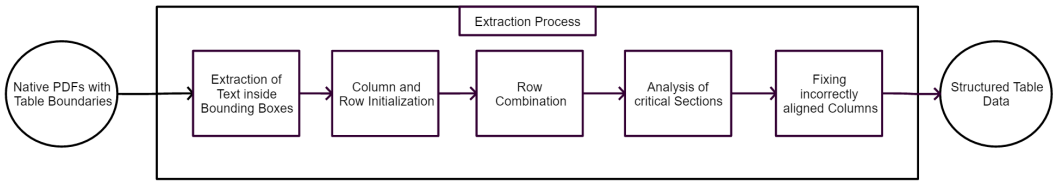


Figure 3.22: The Table Extraction process takes native PDFs as input along with their predicted bounding boxes. The textual content is extracted and structured before the table structure is initialized. Afterwards specific regions are reevaluated and in certain cases restructured before the table data get exported.

For performing the Table Extraction process several in chapter 3.1.2 described algorithms and procedures are reused since the treated documents are from the same type (native PDF documents). The first step after the extraction is initiated is the **extraction of the textual content** of the originally scanned PDFs. Since OCR was performed in the previous phase before the documents were passed to the extraction phase the textual data can be extracted identical to the native approach. Hereby the same methods for text clustering into words, lines and pages are used for being able to handle the input data in a more convenient manner. Also the sparse point minimum value is calculated similar to the native approach since the input available only gives information about the bounding box coordinates of the detected table but not about the row and column structure. This value will allow to structure the table after the content inside the borders is determined. Therefore the input .csv file is analysed line by line and every entry is stored as a table region. When the regions based on the given coordinates (upper left and lower right corner) are determined the affected textual data is assigned to the region.

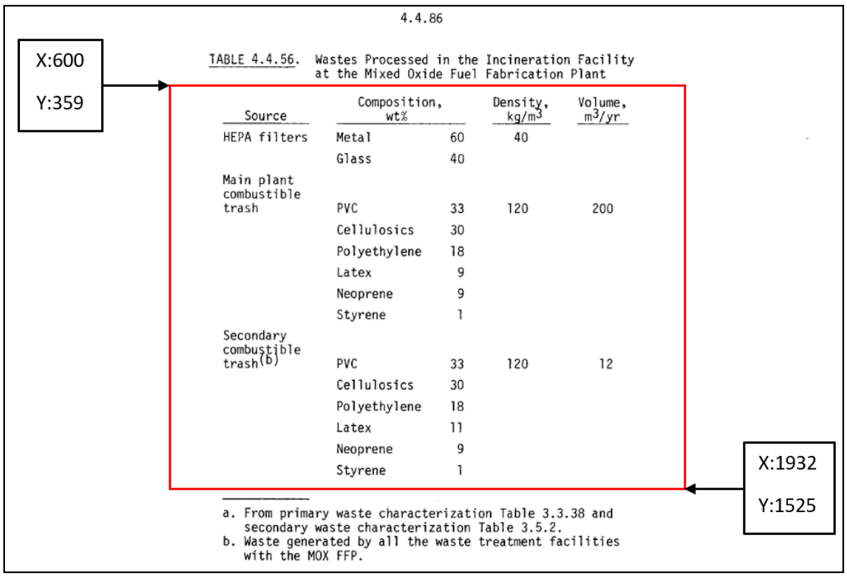


Figure 3.23: Given the coordinates describing the upper left and lower right points of the table bounding boxes the whole table area can be reconstructed. Hereby the textual content situated in between this area is summarized to one tabular structure before further actions are taken.

After all table regions are correctly matched with the textual data situated inside its border the structuring process of the table itself begins. Since there are no differences between the analysed data and the data structure when performing Table Extraction in the native PDF Table Extraction process, the same algorithms and methods are used to do so 2.2.1. After the structure was initialized the rows and columns get assigned their cell values. To identify wrong assignments critical sections are reevaluated incorrect structures are restructured like described in chapter 2.2.1. As a result of this process the table structure gets exported as .csv to allow the user easily go through all detected tables of the document while saving time that would have been necessary to detect and manually copy the tables.

3.2.3 Problems and Challenges

In this chapter problems and challenges that were faced during the development process of the deep learning based Table Extraction tool for scanned PDFs. Com-

pared to the heuristic tool the number of problems is lower but on the other hand the significance of the problems that came up is much higher.

- **Framework Selection**
  To implement the described approach for Table Detection in native PDF documents, a framework to perform deep-learning is necessary. The target of deep-learning frameworks lies in allowing the developer to build models based on an architecture optimized and pre-built for this purpose without having to be concerned about the underlying algorithms. For the development of the scanned Table Detection tool described in this thesis the TensorFlow Framework was chosen. Even though there exist several alternatives (Pytorch, Sonnet, mxnet and many more), it was decided to select TensorFlow for the implementation because of its multi-language-support (Python, C++, R) and the excellent documentation available. The tool is developed by Google and will therefore be further improved and will therefore stay relevant for a while. Moreover CPU as well as GPU support is provided and the TensorBoard component allows to visualize relevant data [13, 12].

- **Model Selection**
  To realize the implementation of the Table Detection tool for scanned PDF documents a deep learning-based approach was selected. To implement the approach, it was necessary to select a model appropriate for the specific case of application. Therefore profound research was conducted to evaluate the available options. Based on the results of the comprehensive analysis the Faster R-CNN model was selected. The Faster R-CNN model is currently the state-of-the-art approach when performing object detection tasks. It stands out due to its excellent performance since a RPN (Region Proposal Network) is used instead of selective search to generate the ”interesting” boxes. Furthermore it was essential that the selected model is supported by the framework (TensorFlow) that is used to implement the deep-learning component.

- **Training Stage**
  - Find appropriate learning rate
    It is hard to find the optimal learning rate because multiple testing cycles have to be conducted and the performance has to be evaluated after the training process has finished for being able to determine the performance of a specific learning rate.

  - Analyze performance of model (validation)
    Model evaluation is also a time-consuming process since after a certain amount of iteration key-parameters have to be compared to the results of the previous run. Like this it can get checked whether the training did improve the results or not.

- **Extraction Stage**
  - Performance impact of OCR
    To perform OCR on the scanned PDF documents the Tesseract engine is used. Even though Tesseract delivers good results the usage of Tesseract also has a drawback. The OCR process is time and resource consuming and slows down the tools execution significantly.
  - Align coordinates after document recomposition
    After the image files are recomposed into native PDFs, these are passed into the work-flow of the java based Table Extraction tool. When extracting the native textual input the original coordinates are differ from the coordinates that are passed in form of .csv files to identify the table bounding boxes. Therefore the coordinates of the extracted textual content have to be adapted in a certain manner to make them fit the given input coordinates of the scanned PDF Table Detection method.

4 Evaluation

This chapter focuses on evaluating the two different methods of Table Detection described in the previous chapter 3. For performing the evaluation, several factors described by Göbel, Oro and Orsi [32] are considered concerning the selection of data and building the ground truth of the selected data set. Also, parameters relevant for the analysis of the performance of a Table Detection tool are chosen concerning relevant structural attributes (single column, multi column, number of pages). The evaluation is based on several statistical values including Recall, Precision and F1-Score.

4.1 Evaluation Criteria

To identify how well the two approaches perform, three metrics, **Recall**, **Precision** and **F1**, are used. These metrics are chosen as they give a good overview about the overall performance of the applied approaches and reflect the completeness of the detected tables, as well as their ability to classify only correct tabular structures. To evaluate the overall performance considering Precision and Recall, the harmonic mean is calculated (F1-Score). In order to calculate the above listed statistical performance indicators, it is necessary to know several numbers that describe the detection results in detail. These relevant parameters are the following:

- Number of tables in PDF documents
- Number of tables detected
- Number of correct tables detected (True Positive = TP)
- Number of incorrect tables detected (False Positives = FP)
- Number of not detected tables

Following Figure 4.1 gives an overview about the above mentioned key parameters. As it can be seen, the whole data is split in two halves, the left half representing

actual tabular data, whereas the right half represents non-tabular data. Identified tables are part of the circle in the center of the rectangle. These identified table regions are categorized according to whether they are actually correctly identified (TP) or they don‘t represent a table in the original document (FP). Data not being categorized as tabular is part of the area outside of the circle. Hereby the False Negative (FN) fraction contains tables that were not detected by the Table Detection algorithm, whereas the True Negative (TN) fraction consists of non-tabular data that was identified as such.

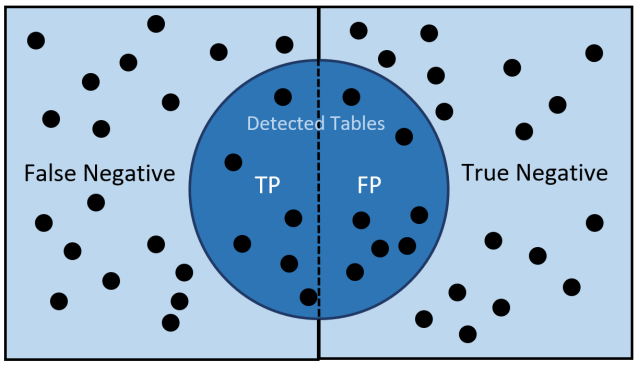


Figure 4.1: This Figure shows the relevant portions of data that have to be considered for calculating the Recall, Precision, as well as the F1 value. The circle represents the detected table portion that is again split into two categories (FP, TP), depending on the real existence of these tables. The points positioned outside the circle represent the data not classified as table. Hereby again two parts are existing, FN being the non-tabular data and FN being the tabular data that was non classified as such.

Recall describes the degree of completeness of the retrieval process. The relation between the found relevant documents (TP) and all found documents (the whole circle in Figure 4.1) is calculated. The Recall value is always in the range between 0 and 1, 0 being the worst and 1 being the perfect result. The formula for calculating the Recall is the following:

$$Recall = \frac{TP}{FN+TP}$$

Precision helps measuring the accuracy of the search process and shows the ability of eliminating non-relevant information. The ratio between the relevant found documents and all found documents is calculated. Again, the results of this calculation

are situated between 0 and 1, 0 being the worst and 1 being the perfect result. To calculate the Precision the following formula is used:

$$Precision = \frac{TP}{FP+TP}$$

According to Buckland and Gey [20], a trade-off between the two values Recall and Precision is unavoidable. The higher the Precision of the detection algorithm, the higher is the chance that some ”Positives” are not detected and categorized as FN. On the other hand, when the Recall value is very high, typically the Precision is reduced since a higher amount of FPs is likely to appear.

The final Parameter used to describe and evaluate the detection results is the **F1-Score**. The F1-Score describes the weighted average of Recall and Precision. If the F1-score is high, the performance of precision and recall will also show good results. The formula used to calculate the F1-score is the following:

$$F1 = 2 * \frac{Precision*Recall}{Precision+Recall}$$

In order to allow proper interpretation of the results, several different scenarios are considered that lead to varying results. This especially concerns tables that were detected partially, since a clear separation between correct and incorrect detection is hard to determine in such cases. Before going on with the description of the test data used for the evaluation procedure, all possible results of the detection phase are described, categorized and outlined in form of figures to give a comprehensible overview about the whole evaluation process.

For the calculation of the described metrics (Recall, Precision, F1-Score) the following result types are interpreted:

- Correct Detections — True Positives
  This category includes correctly identified tabular structures. The bounding boxes include the whole tables and no textual elements that are not part of a table.
- Missing Detections — False Negatives
  This concerns tables that were not recognized by the currently analysed Table Detection approach.

- Incorrect Detections — False Positives
  Areas categorized as tabular structures that do not contain any form of real table are incorrect results.
- Partial Detections — True Positives of False Negatives
  Table structures that were recognized but the extracted boundaries do not fit sufficiently into the original boundaries for them to be classified as correct, are classified as partially correct.

Based on the described categories that are additionally illustrated in Figure 4.2 the following calculations are used to obtain the metrics used to evaluate the performance of each Table Detection tool. Since the corner cases coming along with partial detections are hard to strictly classify as Correct Detections or Missing Detections, the calculations are performed twice, considering both scenarios. Once partially detected tabular structures are classified as correct results and once they are classified as missing.

Partial Detections categorized as Correct Detections:

$$Recall = \frac{CorrectDetections + PartialDetections}{CorrectDetections + PartialDetections + MissingDetections}$$

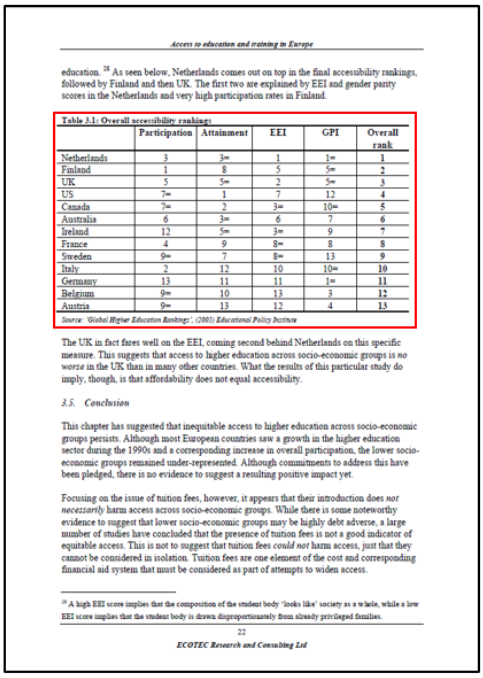
$$Precision = \frac{CorrectDetections + PartialDetections}{CorrectDetections + PartialDetections + IncorrectDetections}$$

Partial Detections categorized as Missing Detections:

$$Recall = \frac{CorrectDetections}{CorrectDetections + MissingDetections + PartialDetections}$$

$$Precision = \frac{CorrectDetections}{CorrectDetections + IncorrectDetections}$$

As stated above, the Recall and Precision of each approach is calculated twice in order to allow an objective evaluation of the approach. To do so, the Partial Detections are added to the Correct Detections at first to compute the performance. Afterwards, the Partial Detections are included in the False Negative category. This way, it is possible to describe the performance considering both evaluation results. This leads to more comprehensible and transparent results that will be described in Chapters 4.4.



Access to education and training in Europe

education.[28] As seen below, Netherlands comes out on top in the final accessibility rankings, followed by Finland and then UK. The first two are explained by EEI and gender parity scores in the Netherlands and very high participation rates in Finland.

Table 3.1: Overall accessibility ranking:

	Participation	Attainment	EEI	GPI	Overall rank
Netherlands	3	3=	1	1=	1
Finland	1	8	5	5=	2
UK	5	5=	2	5=	3
US	7=	1	7	12	4
Canada	7=	2	3=	10=	5
Australia	6	3=	6	7	6
Ireland	12	5=	3=	9	7
France	4	9	8=	8	8
Sweden	9=	7	8=	13	9
Italy	2	12	10	10=	10
Germany	13	11	11	1=	11
Belgium	9=	10	13	3	12
Austria	9=	13	12	4	13

Source: 'Global Higher Education Rankings', (2003) Educational Policy Institute

The UK in fact fares well on the EEI, coming second behind Netherlands on this specific measure. This suggests that access to higher education across socio-economic groups is *no worse* in the UK than in many other countries. What the results of this particular study do imply, though, is that affordability does not equal accessibility.

3.5. Conclusion

This chapter has suggested that inequitable access to higher education across socio-economic groups persists. Although most European countries saw a growth in the higher education sector during the 1990s and a corresponding increase in overall participation, the lower socio-economic groups remained under-represented. Although commitments to address this have been pledged, there is no evidence to suggest a resulting positive impact yet.

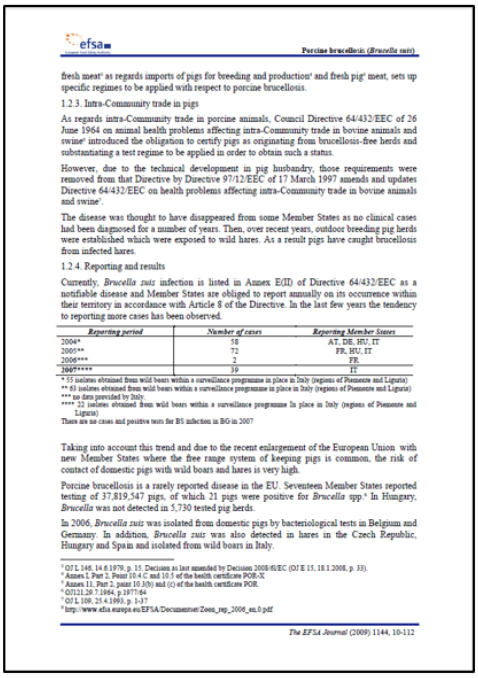
Focusing on the issue of tuition fees, however, it appears that their introduction does *not necessarily* harm access across socio-economic groups. While there is some noteworthy evidence to suggest that lower socio-economic groups may be highly debt adverse, a large number of studies have concluded that the presence of tuition fees is not a good indicator of equitable access. This is not to suggest that tuition fees *could not* harm access, just that they cannot be considered in isolation. Tuition fees are one element of the cost and corresponding financial aid system that must be considered as part of attempts to widen access.

[28] A high EEI score implies that the composition of the student body 'looks like' society as a whole, while a low EEI score implies that the student body is drawn disproportionately from already privileged families.

22

ECOTEC Research and Consulting Ltd

Correct Detection



efsa

Porcine brucellosis (*Brucella suis*)

fresh meat' as regards imports of pigs for breeding and production' and fresh pig' meat, sets up specific regimes to be applied with respect to porcine brucellosis.

1.2.3. Intra-Community trade in pigs

As regards intra-Community trade in porcine animals, Council Directive 64/432/EEC of 26 June 1964 on animal health problems affecting intra-Community trade in bovine animals and swine' introduced the obligation to certify pigs as originating from brucellosis-free herds and substantiating a test regime to be applied in order to obtain such a status.

However, due to the technical development in pig husbandry, those requirements were removed from that Directive by Directive 97/12/EEC of 17 March 1997 amends and updates Directive 64/432/EEC on health problems affecting intra-Community trade in bovine animals and swine'.

The disease was thought to have disappeared from some Member States as no clinical cases had been diagnosed for a number of years. Then, over recent years, outdoor breeding pig herds were established which were exposed to wild hares. As a result pigs have caught brucellosis from infected hares.

1.2.4. Reporting and results

Currently, *Brucella suis* infection is listed in Annex E(II) of Directive 64/432/EEC as a notifiable disease and Member States are obliged to report annually on its occurrence within their territory in accordance with Article 8 of the Directive. In the last few years the tendency to reporting more cases has been observed.

Reporting period	*Number of cases*	*Reporting Member States*
2004*	58	AT, DE, HU, IT
2005**	72	FR, HU, IT
2006***	2	FR
2007****	39	IT

* 55 isolates obtained from wild boars within a surveillance programme in place in Italy (regions of Piemonte and Liguria)
** 63 isolates obtained from wild boars within a surveillance programme in place in Italy (regions of Piemonte and Liguria)
*** no data provided by Italy
**** 22 isolates obtained from wild boars within a surveillance programme in place in Italy (regions of Piemonte and Liguria)
There are no cases and positive tests for BS infection in BG in 2007

Taking into account this trend and due to the recent enlargement of the European Union with new Member States where the free range system of keeping pigs is common, the risk of contact of domestic pigs with wild boars and hares is very high.

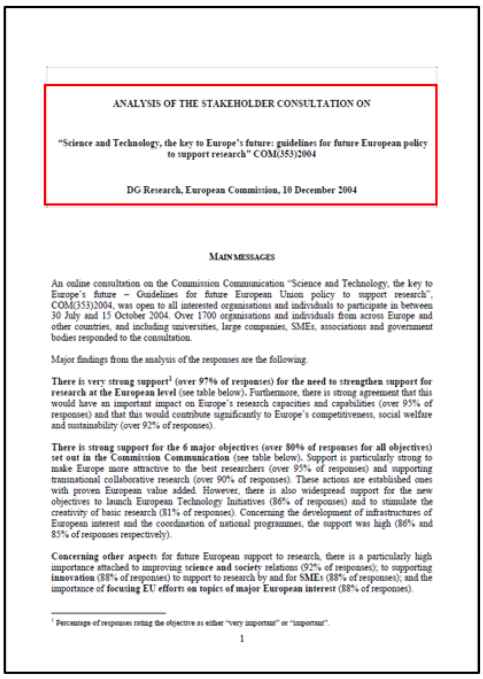
Porcine brucellosis is a rarely reported disease in the EU. Seventeen Member States reported testing of 37,819,547 pigs, of which 21 pigs were positive for *Brucella* spp.[4] In Hungary, *Brucella* was not detected in 5,730 tested pig herds.

In 2006, *Brucella suis* was isolated from domestic pigs by bacteriological tests in Belgium and Germany. In addition, *Brucella suis* was also detected in hares in the Czech Republic, Hungary and Spain and isolated from wild boars in Italy.

[3] OJ L 146, 14.6.1978, p. 15. Decision as last amended by Decision 2008/916/EC (OJ E 15, 18.1.2008, p. 33).
[4] Annex I, Part 1, Point 10.4.C and 10.5 of the health certificate POR-X
[5] Annex 11, Part 2, point 10.3(b) and (c) of the health certificate POR.
[6] OJ 121,29.7.1964, p.1977/64
[7] OJ L 109, 25.4.1993, p. 1-37
[8] http://www.efsa.europa.eu/EFSA/Documentset/Zoon_rep_2006_en,0.pdf

The EFSA Journal (2009) 1144, 10-112

Missing Detection



ANALYSIS OF THE STAKEHOLDER CONSULTATION ON

"Science and Technology, the key to Europe's future: guidelines for future European policy to support research" COM(353)2004

DG Research, European Commission, 10 December 2004

MAIN MESSAGES

An online consultation on the Commission Communication "Science and Technology, the key to Europe's future – Guidelines for future European Union policy to support research", COM(353)2004, was open to all interested organisations and individuals to participate in between 30 July and 15 October 2004. Over 1700 organisations and individuals from across Europe and other countries, and including universities, large companies, SMEs, associations and government bodies responded to the consultation.

Major findings from the analysis of the responses are the following.

There is very strong support[1] **(over 97% of responses) for the need to strengthen support for research at the European level** (see table below). Furthermore, there is strong agreement that this would have an important impact on Europe's research capacities and capabilities (over 95% of responses) and that this would contribute significantly to Europe's competitiveness, social welfare and sustainability (over 92% of responses).

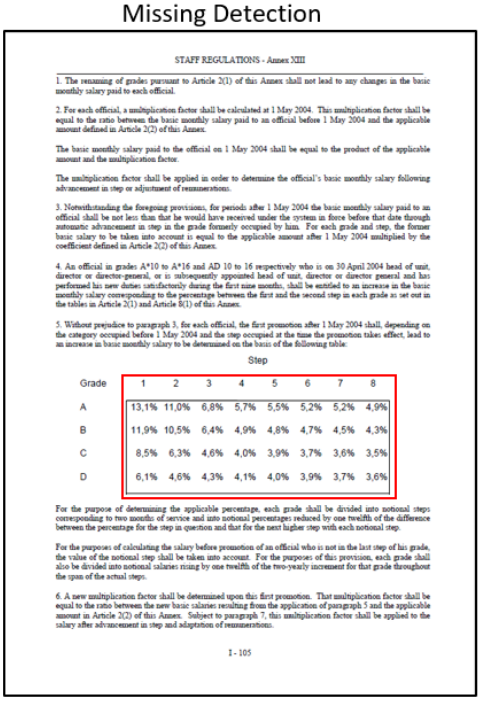
There is strong support for the 6 major objectives (over 80% of responses for all objectives) set out in the Commission Communication (see table below). Support is particularly strong to make Europe more attractive to the best researchers (over 95% of responses) and supporting transnational collaborative research (over 90% of responses). These actions are established ones with proven European value added. However, there is also widespread support for the new objectives to launch European Technology Initiatives (86% of responses) and to stimulate the creativity of basic research (81% of responses). Concerning the development of infrastructures of European interest and the coordination of national programmes, the support was high (86% and 85% of responses respectively).

Concerning other aspects for future European support to research, there is a particularly high importance attached to improving **science and society** relations (92% of responses); to supporting **innovation** (88% of responses) to support to research by and for SMEs (88% of responses); and the importance of **focusing EU efforts on topics of major European interest** (88% of responses).

[1] Percentage of responses rating the objective as either "very important" or "important".

1

Incorrect Detection



STAFF REGULATIONS - Annex XIII

1. The renaming of grades pursuant to Article 2(1) of this Annex shall not lead to any changes in the basic monthly salary paid to each official.

2. For each official, a multiplication factor shall be calculated at 1 May 2004. This multiplication factor shall be equal to the ratio between the basic monthly salary paid to an official before 1 May 2004 and the applicable amount defined in Article 2(2) of this Annex.

The basic monthly salary paid to the official on 1 May 2004 shall be equal to the product of the applicable amount and the multiplication factor.

The multiplication factor shall be applied in order to determine the official's basic monthly salary following advancement in step or adjustment of remunerations.

3. Notwithstanding the foregoing provisions, for periods after 1 May 2004 the basic monthly salary paid to an official shall be not less than that he would have received under the system in force before that date through automatic advancement in step in the grade formerly occupied by him. For each grade and step, the former basic salary to be taken into account is equal to the applicable amount after 1 May 2004 multiplied by the coefficient defined in Article 2(2) of this Annex.

4. An official in grades A*10 to A*16 and AD 10 to 16 respectively who is on 30 April 2004 head of unit, director or director-general, or is subsequently appointed head of unit, director or director general and has performed his new duties satisfactorily during the first nine months, shall be entitled to an increase in the basic monthly salary corresponding to the percentage between the first and the second step in each grade as set out in the tables in Article 2(1) and Article 8(1) of this Annex.

5. Without prejudice to paragraph 3, for each official, the first promotion after 1 May 2004 shall, depending on the category occupied before 1 May 2004 and the step occupied at the time the promotion takes effect, lead to an increase in basic monthly salary to be determined on the basis of the following table:

Grade	Step 1	2	3	4	5	6	7	8
A	13,1%	11,0%	6,8%	5,7%	5,5%	5,2%	5,2%	4,9%
B	11,9%	10,5%	6,4%	4,9%	4,8%	4,7%	4,5%	4,3%
C	8,5%	6,3%	4,6%	4,0%	3,9%	3,7%	3,6%	3,5%
D	6,1%	4,6%	4,3%	4,1%	4,0%	3,9%	3,7%	3,6%

For the purpose of determining the applicable percentage, each grade shall be divided into notional steps corresponding to two months of service and into notional percentages reduced by one twelfth of the difference between the percentage for the step in question and that for the next higher step with each notional step.

For the purposes of calculating the salary before promotion of an official who is not in the last step of his grade, the value of the notional step shall be taken into account. For the purposes of this provision, each grade shall also be divided into notional salaries rising by one twelfth of the two-yearly increment for that grade throughout the span of the actual steps.

6. A new multiplication factor shall be determined upon this first promotion. That multiplication factor shall be equal to the ratio between the new basic salaries resulting from the application of paragraph 5 and the applicable amount in Article 2(2) of this Annex. Subject to paragraph 7, this multiplication factor shall be applied to the salary after advancement in step and adaptation of remunerations.

1 - 105

Partial Detection

Figure 4.2: In this figure the different results that may occur during the Table Detection process are illustrated. This concerns the categories of Correct Detections, Missing Detections, Incorrect Detections, as well as Partial Detections.

4.2 Limitations

Due to the wide range of layouts in which a PDF file can be built, it is hardly possible to implement a heuristic approach that performs on the same level for every type of PDF. For this reason, there are some limitations that have to be mentioned before analyzing the results of the Table Detection method. These limitations also influence the numbers displayed in Table 4.3 and therefore the whole outcome of the evaluation process. Aside from the heuristic approach, the deep learning based approach used for scanned PDF documents is also affected by some limitations. Therefore, the following listing includes the concerned approaches and describes each limitation in detail.

- Documents not containing ”normal” text passages
  Affected: **Heuristic Approach**
  Typically, PDF documents consist of multiple pages containing textual content, images, as well as tables. In rare cases a document only consists of tables without actual textual content the implemented algorithm fails. This is due to the threshold that is calculated to identify sparse areas inside lines. If no text passages are contained in a file and only tables are contained, this threshold calculation provides a value that is not suitable to identify the actual tabular structure afterwards. Therefore, files of this type (typically one page documents) are excluded from the testing data-set.
- Images categorized as tabular structures
  Affected: **Heuristic Approach, Deep Learning based Approach**
  As already mentioned in Chapter 3.1.3, the algorithm struggles in differentiating tables from charts or images containing text. The labels of charts, as well as their textual content, mostly match the criteria that the algorithm uses to identify tables. This is due to the fact that lines included are sparse, as well as successive. Furthermore, the amount of sparse areas are similar for these lines. These circumstances lead to the algorithm to categorize these structures as tables (False Positives) and therefore represents a limitation. Since this limitation is well-known and its impact would falsify the actual results of the table structure recognition process, False Positives appearing due to this, are excluded from the evaluation process and therefore ignored in Table 4.3.
- Incorrect extraction results delivered by third party software
  Affected: **Heuristic Approach**

As already described in Chapter 3.1.3, the heuristic approach depends on external software (in this case PDFBox) that performs the extraction of textual elements of the native input files. If the results that are delivered by the third party library are incorrect, the whole detection fails, since the extracted data forms the basis for the whole process. Even though this limitation hardly ever occurs, it is mentioned to give a complete list of scenarios that may influence the outcome of the evaluation process.

- Pseudo-Code and Formulas
  Affected: **Heuristic Approach, Deep Learning based Approach**
  When considering scientific documents, frequently formulas or even Pseudo-Code (especially in the field of computer science and software engineering) areas appear. In those cases neither the native nor the scanned approach perform well in differentiating between real tables and those mentioned structures. This leads to a accumulation of FPs that will be ignored when performing the calculation of the evaluation metrics, since this weakness is well-known and would falsify the resulting statistics.

This list of limitations describes several scenarios, as well as dependencies, that can have a negative impact on the performance of the discussed Table Detection approaches. Since the described items are very hard to eliminate reliably, those are excluded when evaluating the actual results of the detection process to acquire objective results that describe the performance regardless of document type, document layout and subject area.

4.3 Test Data

In order to perform the evaluation of both table-detection approaches, test data is selected and prepared. The data selected has to cover different document types (multi-column, single-column) in order to reveal strengths and weaknesses that a certain approach has when treating specific file-types. Furthermore, it is essential to select test data that represents ”real-world” PDF documents. By choosing these kind of PDFs, it is possible to evaluate how the analysed approaches perform in practice. For choosing appropriate test data that meets the described criteria, the Google Scholar [1] platform was used and specific search queries were conducted that will

[1] https://scholar.google.at/

be described in detail in the next paragraphs. The data used for test purposes varies for both analysed approaches, since a different type of document is needed as input. In order to evaluate both approaches objectively and to compare them with each other, one part of the data-set is adapted to make it processable for both approaches. Therefore a number of native documents are converted into scanned documents (each page is converted into an image). This way, it is possible to process documents that seem to be the same at first glance for both, the native, as well as the scanned Table Detection approach. The results are afterwards evaluated and interpreted before performing a comparison based on several indicators. In order to gather the test data with the help of the Google Scholar platform, following search queries are used: ”Software Testing”, ”Deep Learning”, ”Automobile Industry”, ”Psychology”, ”Food Industry”, ”Music Industry” and ”Creative Industry” and 34 documents that contain one or more tables are selected. These documents contain 152 tables spread over 589 pages. Data from all these fields is collected to put together a data-set of scientific articles that is close to real-world data because of its high diversity. In addition to the data collected via Google Scholar, there is further test data collected from governmental websites (native). Furthermore, an already prepared data-set is used to evaluate the scanned Table Detection approach. In Chapters 4.3.1 and 4.3.2 this data is described in more detail and an overview over the whole data-set that is finally used to perform the evaluation is given.

4.3.1 Native Test Data

Apart from the already described data-set that is used to evaluate the performance of the scanned, as well as the native Table Detection method, parts of the ICDAR 2013 data-set [2] are utilized to analyse the heuristic approach targeted on native documents. The ICDAR 2013 data-set is an accumulation of PDF documents that are collected from US and EU governmental websites. This data-set includes documents of various structures, including single-column and multi-column PDFs, as well as documents from varying sizes (from one-page documents up to 30+ pages). Every document at least contains one table but there may be pages not containing a single table structure, which increases the probability of False Positives (table recognitions in areas not containing a tabular structure). For the native test data-set the ground truth is manually collected and stored in form of a .csv file. For each

[2] http://www.tamirhassan.com/html/competition.html

table the page and position attribute is stored. This allows to identify tables that are detected correctly (TP), non-tabular areas that were incorrectly categorized (FP) as a table, as well as tables that were no detected (FN). Based on these values, the statistical parameters described in chapter 4.1 can be calculated and an evaluation of the performance of the Table Detection tool for native PDF documents can be conducted.

Type	Documents	Pages	Tables
single-column	59	575	170
multi-column	15	138	56
total	74	713	226

Table 4.1: Amount of native documents being multi column, single column, as well as the total page count for each type is displayed. Additionally, the amount of existing tables for each document type is shown.

4.3.2 Scanned Test Data

The data-set chosen to evaluate the performance of the Table Detection method for scanned PDFs, consists of 34 files of various types as described in Chapter 4.3, in addition to the files that are used to evaluate both approaches. Since prepared data-set only consists of single-page documents, it was necessary to additionally collect a number of multi-page documents to conduct a more objective evaluation, since the evaluation data-set used for the native approach also consists of single-page as well as multi page documents.

The ground truth of the data-set was largely already available since the data-set offered by [10] already provides the necessary information. The information used describing the ground truth for scanned PDF Table Detection is the coordinate information of the table bounding box. Given these coordinates, the results of the detection process can be evaluated and classified as successful or not. Since multi-page documents were added to the evaluation data, the ground truth of those PDF files was created manually using a visual tool that supports the extraction of coordinates in images at certain positions. As the creation of the ground truth can

Type	Documents	Pages	Tables
single-column	61	498	171
multi-column	37	156	97
total	98	654	268

Table 4.2: The amount of scanned documents being multi column, single column as well as the page count for each type is displayed. Also the number of tables is shown for each type and for the total table-count.

be a subjective matter, data that could not be clearly categorized was excluded from the Evaluation data-set as proposed by Göbel, Oro and Orsi [32].

4.4 Results

In this chapter the results of the heuristic and the deep learning based Table Detection processes are presented. Based on the metrics Recall, Precision and F1-Score, the quality of each analysed approach concerning different factors is measured. Hereby, strengths and weaknesses of the heuristic and the deep learning based approach are brought up, considering different styles and types of documents. This concerns the circumstance of whether a document layout consists of one or multiple columns. Furthermore, in order to allow an objective interpretation of the performance, partial detections are interpreted in two different variations (True Positive, False Negative).

4.4.1 Native PDF Table Detection

This section describes the results of the native Table Extraction approach. Hereby, several essential values are considered that will be displayed in the following Table 4.6. The test-data, containing in total 226 tables, is described in Chapter 4.3.1. 170 of these tables are part of a single-column document and 56 are part of a multi-column document. After the detection process the results concerning correct, incorrect, missing and partial detections are collected and structured:

Type	existing Tables	correct	missing	incorrect	partial
single-column	170	154	3	27	13
multi-column	56	46	2	13	8
total	226	200	5	40	21

Table 4.3: The results of the Table Detection process for native PDF documents are shown. Hereby, all existing tables as well as the amount of tables correctly and incorrectly found is relevant. Also, the number of tables that were not detected and the number of missing tables is indicated.

Based on the numbers shown in Table 4.3 the values Recall, Precision and F1-Score (described in Chapter 4.1) can be calculated and interpreted afterwards. Since partially detected tables can be categorized in two different ways (correctly detected = True Positive, not detected = False Negative), the results are calculated twice and the interpretation is based on both of them. In order to give further information about strength and weaknesses of the applied approach, the result-data will be split into single-column and multi-column since it is from interest to see whether the approach is suited better for one of these layouts than for the other one. Finally, the results without considering the document type are shown to give an overview of the overall performance.

The following Table 4.4 shows the Precision, Recall and F1-Score results for the native Table Detection approach, considering single-column and multi-column documents. Since partially detected tabular structures are hard to categorize as correct or incorrect, the calculations are performed twice to allow an objective analysis of the performance.

Partially detected tables are counted as valid detection results (True Positive) in Table 4.4, whereas partially detected tables are counted as invalid detection results (False Negative) in Table 4.5.

As can be seen in Table 4.4, the Recall is very high (total 0.978) when partial results are considered to be correct results. For this reason, the native approach performs well in detecting existing tables for both multi-column, as well as single-column documents. In contrast to that, when partial detections are classified as incorrect, the Recall decreases especially for multi-column documents. While the heuristic approach still performs decently for single-column documents, a slight

Type	Precision	Recall	F1-Score
single-column	0,861	0,982	0,918
multi-column	0,806	0,964	0,878
total	0,847	0,978	0,908

Table 4.4: Native detection results with respect to the document format. Partial Detections are considered as True Positives.

Type	Precision	Recall	F1-Score
single-column	0,851	0,906	0,877
multi-column	0,780	0,821	0,800
total	0,833	0,885	0,858

Table 4.5: Native detection results with respect to the document format. Partial Detections are considered as False Negatives.

performance decrease can be observed when treating multi-column PDFs. Due to the number of Incorrect Detections, the Precision is lower than the Recall. As already mentioned, a trade-off between Precision and Recall is inevitable. Since the positive effect of correctly detected tables outweighs the negative effect of incorrect detections, the Recall was chosen to be from higher relevance than the Precision during the optimization of the results. When comparing the performance decrease when treating partial detections in both ways, it gets obvious that this has a bigger influence on the Recall than on the Precision. While the Precision only decreases slightly (0.01 and 0.026), the Recall goes down much more (0.076 and 0.143). But still the Recall metric holds the superior performance. The weighted average of Precision and Recall (F1-Score) takes into account the Recall as well as the Precision. When analysing the results of the F1-Score calculation, it gets obvious that the heuristic approach performs better for single-column documents than for multi-column documents.This holds true for both cases, especially when considering partial detections to be False Negatives. All in all, the approach performs very solidly for both considered types of documents (single-column and multi-column). The strength of the approach lies in the detection of existing tabular structures, as

represented by the Recall value. Due to the heuristics that are applied to detect table areas, several scenarios that lead to incorrect categorizations of non-tabular areas can appear. This circumstance leads to the Precision of the method not being as high as its Recall value. Generally speaking, in order to detect tables in PDF documents from native nature, this approach is highly effective but in certain scenarios False Positives cannot be avoided.

4.4.2 Scanned PDF Table Detection

This section describes the results of the scanned Table Extraction. Hereby, several essential values are considered that will be displayed in the following Table 4.6. The interpretation of the table rows can be looked up in Chapter 4.1 since both approaches are analysed on the basis of the same result parameters.

Type	existing Tables	correct	not detected	incorrect	partial
single-column	171	156	8	16	7
multi-column	97	87	8	8	2
total	268	243	16	24	9

Table 4.6: The results of the Table Detection process for scanned PDF documents are shown. Hereby, all existing tables, as well as the amount of tables correctly and incorrectly found, is relevant. Also, the number of tables that were not detected is indicated.

The values displayed in Table 4.6 serve as base for the performance evaluation of the deep learning based approach for Table Detection in scanned PDF documents. Given these values the Recall, Precision and F1-Score are calculated presented and analysed in the following paragraphs. In order to eliminate the problem of categorizing partially detected table areas as True Positives or False Negatives, since this is hard to put into practice in a reliable manner, the mentioned values are calculated twice. In Table 4.7 the partially detected tables are categorized as correct and in Table 4.8 they are categorized as incorrect detection results. Additionally, the results are split into single-column and multi-column which allows to recognize areas in which the detection method performs better or worse.

Partially detected tables are counted as valid detection results (True Positive) in Table 4.7, whereas partially detected tables are counted as invalid detection results (False Negative) in Table 4.8.

Type	Precision	Recall	F1-Score
single-column	0,911	0,953	0,931
multi-column	0,918	0,918	0,918
total	0,913	0,940	0,926

Table 4.7: Scanned detection results with respect to the document format. Partial Detections are considered as True Positives.

Type	Precision	Recall	F1-Score
single-column	0,907	0,912	0,910
multi-column	0,916	0,897	0,906
total	0,910	0,907	0,908

Table 4.8: Scanned detection results with respect to the document format. Partial Detections are considered as False Negatives.

The tables above show the result metrics that are calculated based on the detection result values shown in Table 4.6. While the Precision shows nearly no discrepancy between single-column and multi-column documents, the Recall is significantly higher for single-column documents. This can especially be seen in Table 4.7, where partial detections are considered to be correct. Additionally, it can be observed that the Precision drops just by a minimal amount (0.004, 0.002 and 0,003) when taking partial detections into account as False Negatives. This is not the case for the Recall that drops significantly (0.041, 0.021 and 0,033) when comparing both tables. These statistics highlight the ability of the scanned approach to detect tables as a whole. Just a very low percentage of tables are partially detected, which leads to very low variations between Table 4.7 and Table 4.8. All in all, it can be observed that the F1-Score displays better results for the scanned Table Detection approach when treating single-column documents in comparison with multi-column documents.

The distance between both types concerning the performance decreases when partial tables are classified as incorrect.

4.4.3 Comparison of Approaches

After the results of both approaches using different data-sets as test data were collected, another testing cycle is performed to get a deeper understanding of the strengths and weaknesses of each approach. In order to extract results that can be compared for this evaluation cycle, both Table Detection techniques are applied on the same data-set. To do so, the native documents are rendered into scanned PDFs. Like that, the same testing data, consisting of 589 pages including 165 tables, can be used as input for both the scanned and the native approach.

	Native			Scanned		
Type	Precision	Recall	F1-Score	Precision	Recall	F1-Score
single-column	0,867	0,970	0,916	0,901	0,956	0,928
multi-column	0,803	0,961	0,875	0,891	0,961	0,925
total	0,845	0,967	0,902	0,898	0,958	0,927

Table 4.9: Scanned and native detection results with respect to the document format. Partial Detections are considered as True Positives.

	Native			Scanned		
Type	Precision	Recall	F1-Score	Precision	Recall	F1-Score
single-column	0,854	0,871	0,863	0,899	0,939	0,918
multi-column	0,774	0,804	0,788	0,891	0,961	0,925
total	0,827	0,849	0,838	0,897	0,945	0,920

Table 4.10: Scanned and native detection results with respect to the document format. Partial Detections are considered as False Negatives.

The Tables 4.9 and 4.10 illustrate the performance of both Table Detection approaches applied on the same data-set. The initially native PDF data-set was pre-

pared before the scanned approach was applied. Therefore, the pages were rendered to transform every native document into a scanned one. The results show that concerning most metrics the deep learning based approach outperformed the heuristic one. Especially the Precision shows large gaps when considering the multi-column documents. The category in which the native approach is superior to the scanned approach concerns the Recall for single-column documents, when partial detections are considered as valid. The F1-Score displays the overall performance of the tools taking into consideration the Precision as well as the Recall metrics. It can be seen that, overall, the deep learning based implementation dominates every category, independent of the document type and the interpretation of the Partial Detections.

4.5 Discussion

Due to a number of reasons, the problem of table detection in PDF documents is very complex. Documents can be built in different ways, which makes it impossible to handle every PDF the same way. Apart from its generation, the layout varies highly, especially when dealing with documents from different subject areas. Furthermore, there is no blueprint of how tables should be structured and what they should look like. For this reason it is not possible to easily identify heuristics which allow to identify and extract every table in PDF documents. Another challenge when dealing with Table Detection are several structural elements (figures, pseudo-code, formulas), which, due to similar characteristics concerning their appearance, can in certain cases no be differentiated from tabular areas. The purpose of the Discussion Section is to give an overview of the results of both approaches delivered and to compare them to see in which areas the heuristic as well as the deep learning based approach show strengths or weaknesses. Additionally, the research question of this thesis is dealt with by interpreting the results.
In general, both approaches perform solidly when considering the metrics used to evaluate their performances, yet there are still several indicators that show differences in specific scenarios. The Precision of the deep-learning based approach is better than the Precision of the heuristic approach, since the high amount of rules built to detect tables in the heuristic approach often classify non-tabular structures as valid tables. This is not as often the case for the scanned approach,

which leads to much better results concerning Incorrect Detections (False Positives). The number of incorrectly classified table structures is significantly lower for the deep-learning based approach when compared to the heuristic approach. This leads to the conclusion that, for the approach applied on scanned documents, the results are more precise. In the other hand, when considering the Recall of both methods, it is obvious that the heuristic approach performs superior as long as partial detections are considered as correct. If partial detections are classified as False Negatives, the scanned approach performs slightly better for single-column documents and significantly better when treating multi-column documents. Based on these facts, the perception that, especially when performing Table Detection on single-column documents, the heuristic approach performs excellently with a Recall value of 0.982, while the deep learning based approach Recall is 0.953. The drawback of the heuristic approach is the high number of Partial Detections as well as Incorrect Detections in comparison to the method built to detect tables in scanned documents.
In general, the deep learning based approach performs solidly for both types of documents. While the Precision value is very similar no matter which type of document is considered, the Recall shows that, when Partial Detections are considered as correct, the results of Table Detections in single-column documents are more complete. When partial detections are considered as correct, the total Recall is significantly higher, while in Table 4.8 the discrepancy is minimal and in favor of the Precision metric.
In comparison to the scanned approach, the completeness of the detected tables is much higher for the native approach, when considering Partial Detections as correct. The backside of the excellent Recall is the high number of False Positives, as well as Partial Detections, which lead to a significantly lower Precision compared to the deep-learning based approach. The high number of Partial Detections is caused by the different heuristics that are applied to e.g. split tables based on varying line distances. Several factors may cause wrong assumptions, which lead to the classification of non-tabular areas as tables, yet, on the other hand, hardly any table is not recognized as it is reflected by the numbers in Table 4.3. While the native approach only skips 5 tables in total, the approach applied on the scanned documents ignores more than three times the amount of tables (16).
Even within specific scientific fields (like health industry, computer science or automobile industry) not even a rough standardization of the PDF layouts and table structures has been defined. Therefore, it is not possible to give any recommendations concerning the approach that is the best to use for any of these fields. Which

algorithm to choose depends on several factors. Apart from the obvious reasons concerning the document type (scanned, native), both the layout of the documents and its orientation are decisive factors. The heuristic tool often fails when documents that do not contain textual content (files just containing tables) are analysed. This occurs because the threshold calculation for classifying sparse areas in lines fails, since no actual paragraphs containing text are included. Furthermore, the orientation of the documents is essential, since orientation switches in documents can lead to incorrect classifications due to the fact that the extraction process is not optimized to handle this scenario. On the other hand, uncommon layouts of PDF documents or tables may lead to incorrect results for the deep-learning based approach, since the used model is just trained on a limited amount of files that don‘t cover every possible layout a PDF document can have. When interpreting Tables 4.9 and 4.10, which demonstrate the performance of both approaches applied on the same data set, it gets obvious that, especially concerning the Precision metric, the deep learning based approach performs superior to the heuristic approach. Nevertheless, the heuristic approach does a good job at identifying table structures (especially in single column documents), which is reflected by the high Recall. Furthermore, for Table Detection in multi-column documents, the scanned approach provides better results.

5 Conclusion

The PDF standard still is one of the most dominant file formats used for information exchange and representation. Even though at first glance PDF documents may look the same, there are different ways in which they can get created. Depending on the creation-type (directly from applications, calling OS printing API or third party translation software), a different set of information is available for further processing after the creation. The main categories in which PDFs can be divided are native as well as scanned PDF documents. While native PDFs still contain high-level information in different forms, scanned PDFs are image only without including any digital code to describe structure of formatting. Furthermore, different layouts (single-column, multi-column) and orientations (horizontal, vertical) make the topic of Information Extraction from PDF documents very challenging. Since textual content cannot be extracted from scanned documents without further processing, Optical Character Recognition has to be conducted to convert image-typed content into a format a computer can process. Since tables can be found in nearly every type of PDF document and the information density in tables is particularly high, the field of Table Extraction is of high relevance. Even though there are multiple tools available to the public, it is hard to choose the suitable one in specific situations. Additionally, most publicly available tools are limited on Table Detection in native PDF documents, which reduces the flexibility for the user drastically.
To develop a strategy on when the usage of a tool specialized on scanned documents is beneficial compared to a tool targeting native PDFs, two tools for Table Detection are developed and analysed. While the approach targeting native PDF documents is heuristic, the second approach for detecting table structures in scanned PDFs is based on deep-learning. The heuristic method is split in the processing steps ”Preprocessing”, ”Table Detection” and ”Table Extraction”. The applied heuristics were developed by analysing similarities in table structures and how to distinguish them from non-tabular content. The key rule concerning the detection of candidate table lines are sparse fields in between words that are situated in the same line. If a certain threshold is exceeded, the algorithm marks a line as sparse and afterwards

consecutive sparse lines are joined together to build a tabular structure. As model used for the deep learning based approach for scanned documents, the Faster R-CNN algorithm, which is optimized for real-time object detection, was selected. Before the detection of the bounding boxes representing the table structures on the image-typed pages is conducted, every page is preprocessed by applying image transformation algorithms. This is done because the Faster R-CNN object detection method performs best on natural images. By applying image transformation techniques, the input images are converted into images that are as close as possible to natural images. The results of the Table Detection on scanned documents are represented by coordinates values which hold the boundary-values of the detected tables. Since the text extraction is not possible from image typed documents, Optical Character Recognition has to be applied in advance. After the detection phase is completed and the actual table structure is built (rows, columns) and the table-cells are filled with information, Header and Footer detection is initiated based on key words and line distances. The final result of the Table Extraction, regardless of the applied method, are .csv files containing a worksheet per detected table and the actual table structure as content.

The problems occurring during the development of the heuristic approach were primarily associated with varying document and table layouts. Due to the fact that automated extraction of structured text elements is not possible, words and lines have to be built based on the coordinates of the extracted characters. Furthermore, for extraction purposes a third party library was used. This leads to a factor of dependence on an external tool, which can have negative effects on the overall approach. Furthermore, heuristics have to be developed for several scenarios (e.g. separation of consecutive tables, exclusion of page header/footer).

Challenges in the context of the development of the scanned Table Detection approach concern the selection of a suitable framework, model selection, as well as finding an appropriate learning rate and validating the current performance of the model. Additionally, the OCR process is very resource consuming and is therefore slowing down the execution time significantly.

The evaluation of both approaches based on the metrics Recall, Precision and F1-Score shows that the heuristic approach performs excellent considering the amount of tables not recognized. Therefore, the Recall metric is significantly higher than for the deep learning based approach as long as the partial detections are categorized as correct. On the backside the Precision of the heuristic approach is lower since a larger number of Incorrect Classifications are present. When focusing on the different types of layouts, it can be seen that in general the Table Detection performs better

for single-column documents compared to multi-column documents. Additionally, documents consisting of pages that just hold tables and no textual content, are not suitable for the heuristic approach, since the sparse threshold calculation produces results that are not sufficient for its purposes.
All in all, the type, layout and orientation are decisive factors when choosing the appropriate tool for Table Detection. In general it can be said that if the number of False Positives is irrelevant the heuristic approach is the tool of choice especially when treating single column documents, since the Recall is significantly higher as long as Partial Detections are classified as correct results. The heuristic method also stands out because of its high recognition-speed compared to the slow execution of the deep learning based tool that performs complex, resource-intensive operations. If the Precision is considered to be of higher relevance, the deep learning based approach provides superior results, regardless of the document layout. The overall performance reflected by the F1-Score shows that the algorithm applied on scanned PDFs performs superior taking into consideration single-column as well as multi-column documents. Especially, when the target data-set consists of both types of documents and the factors time as well as resources are irrelevant, the deep learning based approach is the best option. Since the PDF format is not standardized and the layout as well as the type can strongly vary, even within the same scientific field, it is hard choose the perfect tool in every situation, but given the mentioned aspects the selection process is simplified.

5.1 Future Work

The described approaches in this thesis come along with several limitations, especially concerning structures that are incorrectly classified as table-structure regardless of the used methodology (heuristic, deep learning based). This especially concerns Figures as well as Pseudo-Code and Formulas because these structures often share similarities with actual table structures concerning its structural features as well as sparse nature. Heuristics to exclude incorrect results coming along with the mentioned structures would significantly increase the performance of the Table Detection process. Furthermore, a combined approach using the advantages of both approaches and automatically choosing the one optimized for the current scenario, would further enhance the performance as well as usability of the Table

Detection approaches since the users would not be forced to choose by themselves which approach is suitable.

Bibliography

[1] About. https://opencv.org/about/. (Accessed on 08/02/2019).

[2] Do we really need gpu for deep learning? - cpu vs gpu. https://medium.com/@shachishah.ce/do-we-really-need-gpu-for-deep-learning-47042c02efe2. (Accessed on 08/01/2019).

[3] How to configure the learning rate when training deep learning neural networks. https://machinelearningmastery.com/learning-rate-for-deep-learning-neural-networks/. (Accessed on 08/04/2019).

[4] Installation — luminoth 0.2.4.dev documentation. https://luminoth.readthedocs.io/en/latest/usage/installation.html#before-you-start. (Accessed on 08/01/2019).

[5] Pdf techniques — techniques for wcag 2.0. https://www.w3.org/TR/WCAG20-TECHS/pdf. (Accessed on 08/26/2019).

[6] pdf2image · pypi. https://pypi.org/project/pdf2image/. (Accessed on 08/03/2019).

[7] Pycharm: the python ide for professional developers by jetbrains. https://www.jetbrains.com/pycharm/. (Accessed on 08/04/2019).

[8] R-cnn, fast r-cnn, faster r-cnn, yolo — object detection algorithms. https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e. (Accessed on 07/31/2019).

[9] Region proposal network (rpn) — backbone of faster r-cnn. https://medium.com/egen/region-proposal-network-rpn-backbone-of-faster-r-cnn-4a744a38d7f9. (Accessed on 07/31/2019).

[10] Table detection using deep learning - good audience. https://blog.goodaudience.com/table-detection-using-deep-learning-7182918d778. (Accessed on 08/02/2019).

[11] Tensorflow records? what they are and how to use them. https://medium.com/mostly-ai/tensorflow-records-what-they-are-and-how-to-use-them-c46bc4bbb564. (Accessed on 08/02/2019).

[12] Top 10 best deep learning frameworks in 2019 - towards data science. https://towardsdatascience.com/top-10-best-deep-learning-frameworks-in-2019-5ccb90ea6de. (Accessed on 08/23/2019).

[13] Top 5 deep learning frameworks, their applications, and comparisons! https://www.analyticsvidhya.com/blog/2019/03/deep-learning-frameworks-comparison/. (Accessed on 08/23/2019).

[14] Understand the softmax function in minutes - data science bootcamp - medium. https://medium.com/data-science-bootcamp/understand-the-softmax-function-in-minutes-f3a59641e86d. (Accessed on 07/31/2019).

[15] Understanding learning rates and how it improves performance in deep learning. https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10. (Accessed on 08/04/2019).

[16] Using tfrecords and tf.example — tensorflow core — tensorflow. https://www.tensorflow.org/tutorials/load_data/tf_records. (Accessed on 08/02/2019).

[17] What is tensorflow? — opensource.com. https://opensource.com/article/17/11/intro-tensorflow. (Accessed on 08/01/2019).

[18] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[19] H. Bast and C. Korzen. A benchmark and evaluation for text extraction from pdf. In *Proceedings of the 17th ACM/IEEE Joint Conference on Digital Libraries*, pages 99–108. IEEE Press, 2017.

[20] M. Buckland and F. Gey. The relationship between recall and precision. *Journal of the American society for information science*, 45(1):12–19, 1994.

[21] D. D. A. Bui, G. Del Fiol, J. F. Hurdle, and S. Jonnalagadda. Extractive text summarization system to aid data extraction from full text in systematic review development. *Journal of biomedical informatics*, 64:265–272, 2016.

[22] D. D. A. Bui, G. Del Fiol, and S. Jonnalagadda. Pdf text classification to leverage information extraction from publication reports. *Journal of biomedical informatics*, 61:141–148, 2016.

[23] A. Chaudhuri, K. Mandaviya, P. Badelia, S. K. Ghosh, et al. *Optical character recognition systems for different languages with soft computing*. Springer, 2017.

[24] A. Cheung, M. Bennamoun, and N. W. Bergmann. An arabic optical character recognition system using recognition-based segmentation. *Pattern recognition*, 34(2):215–233, 2001.

[25] A. S. Corrêa and P.-O. Zander. Unleashing tabular content to open data: A survey on pdf table extraction methods and tools. In *Proceedings of the 18th Annual International Conference on Digital Government Research*, pages 54–63. ACM, 2017.

[26] J. Damerow, B. E. Peirson, and M. D. Laubichler. The giles ecosystem–storage, text extraction, and ocr of documents. *Journal of Open Research Software*, 5(1), 2017.

[27] S. Deivalakshmi. A simple system for table extraction irrespective of boundary thickness and removal of detected spurious lines. In *Inventive Computing and Informatics (ICICI), International Conference on*, pages 69–75. IEEE, 2017.

[28] D. W. Embley, M. Hurst, D. Lopresti, and G. Nagy. Table-processing paradigms: a research survey. *International Journal of Document Analysis and Recognition (IJDAR)*, 8(2-3):66–86, 2006.

[29] D. Ferrés, H. Saggion, F. Ronzano, and À. Bravo Serrano. Pdfdigest: an adaptable layout-aware pdf-to-xml textual content extractor for scientific articles. In *Language Resources and Evaluation Conference (LREC) 2018; 2018 May 7-12; Miyazaki, Japan.*, 2018.

[30] M. Frey and R. Kern. Efficient table annotation for digital articles. *D-Lib Magazine*, 21(11/12), 2015.

[31] A. Gilani, S. R. Qasim, I. Malik, and F. Shafait. Table detection using deep learning. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 771–776. IEEE, 2017.

[32] M. Göbel, T. Hassan, E. Oro, and G. Orsi. A methodology for evaluating algorithms for table understanding in pdf documents. In *Proceedings of the 2012 ACM symposium on Document engineering*, pages 45–48. ACM, 2012.

[33] L. Group. Know the difference between a native pdf and a scanned pdf. `https://leinickegroup.com/2017/07/31/know-the-difference-between-a-native-and-a-scanned-pdf/`. (Accessed on 02/11/2019).

[34] T. Hassan and R. Baumgartner. Table recognition and understanding from pdf files. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, volume 2, pages 1143–1147. IEEE, 2007.

[35] T. Huynh-Van, K. Nguyen-An, T. L. B. Khanh, H.-J. Yang, T. A. Tran, and S.-H. Kim. Learning to detect tables in document images using line and text information. In *Proceedings of the 2nd International Conference on Machine Learning and Soft Computing*, pages 151–155. ACM, 2018.

[36] A. S. Incorporated. *PDF Reference, sixth edition: Adobe Portable Document Format version 1.7*. Adobe Systems Incorporated, 2006.

[37] N. Islam, Z. Islam, and N. Noor. A survey on optical character recognition system. *arXiv preprint arXiv:1710.05703*, 2017.

[38] R. Jana, A. R. Chowdhury, and M. Islam. Optical character recognition from text image. *International Journal of Computer Applications Technology and Research*, 3(4):239–243.

[39] S. Khusro, A. Latif, and I. Ullah. On methods and tools of table detection, extraction and annotation in pdf documents. *Journal of information science*, 41(1):41–57, 2015.

[40] S. Klampfl, K. Jack, and R. Kern. A comparison of two unsupervised table recognition methods from digital scientific articles. *D-Lib Magazine*, 20(11):7, 2014.

[41] I. Y. Korneev, S. G. Popov, A. S. Makushev, and N. Kolodkina. Retention of content in converted documents, Oct. 1 2015. US Patent App. 14/570,088.

[42] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[43] M. Li, L. Cui, S. Huang, F. Wei, M. Zhou, and Z. Li. Tablebank: Table benchmark for image-based table detection and recognition. *arXiv preprint arXiv:1903.01949*, 2019.

[44] Y. Liu, P. Mitra, and C. L. Giles. Identifying table boundaries in digital documents via sparse line detection. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 1311–1320. ACM, 2008.

[45] M. Masum, S. Kosaraju, T. Bayramoglu, G. Modgil, and M. Kang. Automatic knowledge extraction from ocr documents using hierarchical document analysis. 2018.

[46] A. Nazemi, I. Murray, and D. A. McMeekin. Practical segmentation methods for logical and geometric layout analysis to improve scanned pdf accessibility to vision impaired. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 7(4):23–26, 2014.

[47] C. Patel, A. Patel, and D. Patel. Optical character recognition by open source ocr tool tesseract: A case study. *International Journal of Computer Applications*, 55(10), 2012.

[48] M. O. Perez-Arriaga, T. Estrada, and S. Abad-Mota. Tao: System for table detection and extraction from pdf documents. In *FLAIRS Conference*, pages 591–596, 2016.

[49] V. Ranka, S. Patil, S. Patni, T. Raut, K. Mehrotra, and M. K. Gupta. Automatic table detection and retention from scanned document images via analysis of structural information. In *Image Information Processing (ICIIP), 2017 Fourth International Conference on*, pages 1–6. IEEE, 2017.

[50] R. Rastan, H.-Y. Paik, J. Shepherd, S. H. Ryu, and A. Beheshti. Texus: Table extraction system for pdf documents. In *Australasian Database Conference*, pages 345–349. Springer, 2018.

[51] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[52] T. A. Tran, H. T. Tran, I. S. Na, G. S. Lee, H. J. Yang, and S. H. Kim. A mixture model using random rotation bounding box to detect table region in document image. *Journal of Visual Communication and Image Representation*, 39:196–208, 2016.

[53] B. Yildiz, K. Kaiser, and S. Miksch. pdf2table: A method to extract table information from pdf files. In *IICAI*, pages 1773–1785, 2005.

Appendix