Recognising the contents in digitised financial documents

Simas Rimašauskas, Igoris Belovas

Vilnius University, Faculty of Mathematics and Informatics,
Didlaukio g. 47, Vilnius, Lithuania
simas.rimasauskas@mif.stud.vu.lt

Abstract. The necessity of content recognition in digital documents is ever-increasing in the financial sector. Extracted data is used for fundamental analysis, modelling and portfolio selection. In the most prominent markets, there is a wide array of available sources to obtain the data, such as SEC filings easily. However, it is not so in markets with less investor interest, such as the CEE region or Latin America. Often, the only sources containing the data are primary reports by the company itself. Scarce secondary sources may provide data of dubious reliability. This leads to an excessive workload for analysts, implying the necessity to adapt existing intelligent methods for processing financial data.

Keywords: machine learning, natural language processing, optical character recognition, text recognition, table recognition.

1 Introduction

Smaller, less popular markets often suffer from a scarcity of secondary data sources. Financial analysts focusing on these regions, in turn, depend heavily on primary documents issued by companies. Such reliance leads to a workload bottleneck due to the manual, time-intensive process of extracting valuable data from unstructured or poorly formatted documents. Implementing automatized, intelligent methods to recognize and extract contents could mitigate this problem. Hence, we sought to thoroughly review the underlying algorithms of select machine learning (ML) and natural language processing (NLP) technologies and apply them in experiments conducted on a real-life dataset comprised of quarterly and annual reports from firms operating in exotic markets.

2 Historical development of content recognition

Various approaches for detecting tables in images were devised throughout the years - Dengel et al. [1] proposed a method to cluster word boundaries

DOI: https://doi.org/10.15388/LMITT.2025.22

into a segmentation graph for table detection. However, that failed in the face of multi-column layouts. Wangt et al. [2] proposed using the distance between consecutive words as a heuristic in determining table entity candidates. However, such a method is inevitably tied to a specific layout template, which hinders its practical application. Gatos et al. [3] proposed using the intersection area between horizontal and vertical lines to reconstruct the intersectional pairs. However, this system was said to overly rely on the visual cues provided by strict, defining table borders. Ultimately, deep learning methods were implemented when Hao et al. [4] proposed a regional proposal network using CNNs, which later adapted the faster R-CNN architecture to segment table regions in a given image. Building on this, Watson and Liu proposed a table alignment process involving organizing disjoint text segments into columns by creating disjoint sets and merging them using an algorithm.

A research survey conducted in a paper by Kasem et al. [5] has established that ML and deep learning technologies have been effective in computer vision research tasks such as object detection and object position estimation. CNN, one of the most widely-used types of neural networks, can use the fundamental properties of actual signals, such as translation invariance and compositional hierarchies. A typical CNN comprises a hierarchical structure and numerous layers for learning data representations at different levels of abstraction [6]. The efficacy of CNNs in object identification, the researchers assert, is based on the ability to, through training on large amounts of data, learn substantial mid-level visual properties, which are more complex patterns such as textures and shapes, rather than hand-crafted low-level representations, such as edge detectors, often used in image categorization. Deep learning models are now widely used in multiple areas, including general table detection [4]. On the other hand, table structures receive far less attention, and the table structure is typically only characterized by the rows and columns of a table [7].

3 Summary of state-of-the-art algorithms

In this section, we survey modern NLP and ML technologies and review and describe the subtleties and principles behind their algorithms. Owing to their widespread use in practice, the tools selected for the work are LayoutLM, OpenCV, Camelot, pdfplumber, and Google Document AI.

In their 2020 work, Xu et al. [8] proposed LayoutLM, a pre-training technique with text and layout information in a single framework that utilizes Transformer architecture as the backbone. It is inspired by the BERT model, which represents text using embeddings. According to the researchers, LayoutLM extends the principle by using multimodal inputs, including token, layout, and image embeddings. For the experiments, the LayoutLMv3 version was used. The model works by processing text using an OCR toolkit to extract content and 2D positional information. Word embeddings are then initialized using a pre-trained RoBERTa model. The layout positions are applied at the segment level, where words in the same segment share the same 2D position, as opposed to word-level positions used in previous versions like LayoutLM and LayoutLMv2. Image embeddings are generated by applying linear projections to image patches, which helps to reduce computational complexity compared to previous versions that used CNNs, like Faster R-CNN. Images are then resized, split into patches, and flattened into a sequence of vectors. Semantic 1D relative position and spatial 2D relative position are introduced as bias terms in self-attention networks for both text and image modalities, which follows the previous approach in LayoutLMv2.

Another tool used for data extraction in financial documents is OpenCV, a real-time computer vision programming function library initially launched by Intel in 1999 [9]. The tool provides numerous image manipulation, analysis, and data extraction methods. Mat class serves as the main data structure in OpenCV for the storage and manipulation of images and matrices. It holds pixel values and handles memory management by automatically allocating memory as necessary. It also supports reference counting, allowing efficient memory use by preventing redundant copying of image data when objects are passed by reference. Histograms are commonly used for image processing, representing the distribution of pixel intensity in an image. Additionally, OpenCV supports techniques such as Fourier transform to analyse the frequency content of an image. By decomposing an image into its frequency components, algorithms can be designed to filter out unwanted noise or enhance certain features. Hough transform is also implemented in OpenCV and is used to detect geometric shapes in an image. OpenCV also provides Edge detection as another fundamental image processing technique - a canny edge detector combines gradient calculation with non-maximum suppression to detect edges accurately while reducing noise.

Pdfplumber is an open-source Python library created to extract information from PDF documents. It was created by Singer-Vine and was initially released in 2016 [10]. Pdfplumber builds on top of pdfminer, another PDF processing library. The approach is heavily inspired by Nurminen's master's thesis [11]. In essence, for any PDF page, either actual lines, which are explicitly defined and give the table a grid-like structure, or invisible lines implied by word alignment are found. Overlapping or nearly-overlapping lines are merged, and intersections of all these lines are found. The most granular set of rectangles, which use the intersections as vertices, are found, and contiguous cells are then grouped into tables. Pdfplumber recognizes lines in a PDF by parsing the document's raw content streams to identify vector graphics commands corresponding to line-drawing operations. These instructions are then decoded by extracting the coordinates, orientation, and thickness of lines explicitly defined in the document. The extracted line objects are then interpreted, and the positioning of the text element is examined further to infer implied lines where no explicit borders are present. Text alignment and spacing are analysed to identify patterns that might imply grid structures.

Camelot, a Python library for data extraction from PDFs, was created by Vinayak Mehta [12]. Camelot offers two table parsing methods, Stream and Lattice, each designed to accommodate different PDF table structures. The Stream method, best used for tables that do not have visible borders but instead rely on whitespace between cells to create a table structure, works by grouping words on the PDF page into rows based on their y-axis overlaps. It then calculates text edges to identify areas that could contain tables. The number of columns is estimated by calculating the mode of the word count in text rows, thus determining column ranges. The words are then assigned to columns based on whether they fall within the calculated x and y ranges. The lattice method is more suitable for tables with visible cell borders, relying on precise line detection and geometric calculations. It works by converting the page into an image using Ghostscript. The algorithm then detects horizontal and vertical line segments by applying transformations such as erosion and dilation. The detected lines are processed to identify intersections, marking the boundaries of table cells. They are scaled and translated back to the file's coordinate space to create a representation of the table. Spanning cells are then identified, and the words are assigned to the correct table cells based on their x and y coordinates.

The Document AI platform is a unified document processing console launched by Google Cloud, the cloud computing division of Google, in November 2020 [13]. The platform aims to automate and validate documents by providing access to document parsers, tools, and solutions via an API. Because of the proprietary nature of the application, the table extraction algorithms used by Google Document AI are not disclosed to the public and, therefore, were not reviewed in our work.

The previously described table data extraction methods differ in some ways yet have similarities. Regarding the data extraction approach, pdfplumber and Camelot are similar because both libraries do rule-based parsing, where predefined rules are used to identify and extract specific structures or patterns from documents. OpenCV operates utilizing the approach of low-level image processing, while LayoutLM uses a multimodal transformer architecture that integrates text and visual embeddings for document understanding. LayoutLMv3 and Google Document AI are better suited for complex documents with rich layouts and can handle visual and textual data. Camelot and pdfplumber rely on line detection for grid-based structures, although Camelot's Stream method is optimized for whitespace-based tables, while Lattice is tailored for bordered table layouts. OpenCV uses pixel-level manipulation (such as detection of edges or contour analysis) to identify lines or shapes.

4 Experiments

An original dataset consisting of 7 selected PDF format financial documents published by different companies in Central and Eastern Europe and Latin America was compiled and used for our experiments. The files were quarterly or annual reports published in financial periods from 2018 to 2022. The eight selected tables consisted of the main parts of financial statements (statements of income and loss, financial position, and cash flows), notes, and appendices and were in English. Using a Python API, a ground truth version of the tables and their text content was extracted and reviewed manually.

In order to test the selected tools, five scripts were written in Python, each using a different tool to recognize text and extract tables from PDF pages into a spreadsheet. For LayoutLMv3, the FUNSD dataset was used for the training process, which focuses on form understanding [14]. However,

the model cannot perform text recognition, so the Tesseract OCR engine was additionally integrated, which renders high-resolution images of PDF format file pages and performs optical character recognition (OCR) to detect text and bounding boxes [15]. This consideration also applies to OpenCV. Since it does not perform OCR, Tesseract was also used to extract text and bounding box information from the document images in the OpenCV implementation. Custom functions were developed to merge extracted text boxes when they were sufficiently close in terms of coordinates, improving the coherence of detected text blocks. For Google Document AI, a Form Parser processor was selected to process the document, which was developed for the extraction of tables and is best suited for table data recognition and extraction [16]. Pdfplumber, a library developed explicitly for text extraction from searchable .pdf files and table extraction, offered ample table extraction settings. Because most tables in the dataset were borderless, we used vertical and horizontal strategy values of "text" for the pdfplumber function *extract_tables()*, which allowed the library to aim specifically to identify columns in borderless tables. In the implementation of Camelot, another Python library specifically made to extract tables, there was also little need for pre-processing as the PDF files were simply processed using a library function *read_pdf()*. The previously discussed Stream method, which is more suitable for tables without explicit borders, was chosen. Finally, algorithms were created to evaluate the tables using the defined metrics.

To assess the accuracy of the selected tools, three metrics were defined:

1. **Symbol accuracy** is defined as correctly output characters divided by all output symbols. It was evaluated by comparing the lists of characters for each cell to those from the ground truth.

$$Symbol\ Accuracy = \frac{Number\ of\ Correctly\ Output\ Symbols}{Total\ Number\ of\ Output\ Symbols}$$

2. **Word accuracy** is defined as the proportion of correctly output words (which we define as ordered lists of characters divided by blank characters) to the total number of output words in the table. This and the symbol accuracy metrics are based on the metrics used by Rupšys in his bachelor thesis [17]

$$Word\ Accuracy = \frac{Number\ of\ Correctly\ Output\ Words}{Total\ Number\ of\ Output\ Words}$$

3. **Table accuracy** metric to evaluate the **structural integrity** of tables has been established based on the content accuracy metric used by Smock et al. [18]. By calculating the share of correct cells, we can deduce whether the output table retained the correct structure of the original.

$$Table\ Accuracy = \frac{Number\ of\ Cells\ in\ Correct\ Relative\ Position}{Total\ Number\ of\ Cells}$$

It was found that, regarding the accuracy of the table structure, Camelot strongly outperformed the peer group with a simple average accuracy of 90.0%, as seen in table 1. Google Document AI also provided tables that may be considered satisfactory in terms of structure at 72.8%. LayoutLM resulted in the least accurate table structure at 34.3%. In terms of symbol accuracy seen in table 2, Camelot was strongest among the peer group as well, with a simple average accuracy of 93.0%; Google Document AI also outputs sufficiently accurate tables in terms of symbols at 81.3%. LayoutLM resulted in the least accurate table structure at 23.3%. Lastly, as seen in table 3, the word accuracy is highly correlated with the symbol metric: Camelot performed best in terms of correct words with a 92.1% accuracy. Tables obtained using LayoutLM, which was combined with Tesseract for optical character recognition, provided the lowest average word accuracy at 21.0%.

Table 1. Percentages of identical cells (table accuracy).

Table ID	Total Cells	Camelot	Google	LayoutLM	OpenCV	Pdfplumber
1	165	97.0%	44.2%	57.6%	79.4%	91.8%
2	161	84.5%	87.0%	44.1%	17.4%	77.0%
3	136	98.5%	84.6%	13.3%	72.8%	32.5%
4	96	84.0%	59.4%	44.8%	48.0%	68.0%
5	96	93.8%	93.8%	53.1%	74.0%	5.8%
6	252	97.2%	89.3%	42.6%	24.6%	15.5%
7	168	88.7%	57.7%	14.9%	62.5%	48.8%
8	84	76.2%	66.7%	3.6%	51.2%	21.4%
Average	-	90.0%	72.8%	34.3%	53.7%	45.1%

Table 2. Percentages of correctly output symbols.

Table ID	Total Cells	Camelot	Google	LayoutLM	OpenCV	Pdfplumber
1	165	98.6%	58.9%	36.0%	91.8%	94.5%
2	161	83.9%	92.9%	38.7%	35.7%	71.3%
3	136	96.8%	89.1%	3.2%	79.7%	58.4%
4	96	86.8%	66.9%	20.3%	52.6%	52.4%
5	96	93.2%	87.1%	35.2%	86.1%	5.9%
6	252	98.0%	88.4%	22.5%	42.0%	7.6%
7	168	89.9%	74.0%	14.5%	67.4%	63.5%
8	84	96.5%	93.1%	16.3%	85.1%	40.6%
Average	-	93.0%	81.3%	23.3%	67.5%	49.3%

Table 3. Percentages of correctly output words.

Table ID	Total Cells	Camelot	Google	LayoutLM	OpenCV	Pdfplumber
1	165	97.4%	54.9%	39.4%	81.9%	93.3%
2	161	81.1%	90.6%	23.2%	31.3%	69.5%
3	136	96.3%	83.3%	0.5%	69.9%	45.4%
4	96	85.5%	65.9%	21.0%	50.7%	54.4%
5	96	91.2%	86.1%	38.7%	77.4%	0.0%
6	252	96.9%	86.9%	19.8%	22.8%	0.0%
7	168	92.3%	72.1%	12.4%	64.7%	58.1%
8	84	96.0%	88.5%	13.0%	80.0%	37.5%
Average	-	92.1%	78.5%	21.0%	59.8%	44.8%

5 Conclusion

The overview of the state-of-the-art literature on the topics of NLP and ML in table data extraction, taken together with our practical experiments, has led us to conclude that while deep learning and ML are widely used in table detection, the research on table structures, in particular, has yet to receive wide attention. Regarding the tools used in the experiment, the setup for pdfplumber, Camelot, and Google Document AI is relatively straightforward and quick. Camelot and pdfplumber were the easiest to implement in practice. In contrast, LayoutLM and OpenCV demand more effort and are

time-intensive because of requirements such as model training. In terms of table structure and symbol and word accuracy, Camelot performed the task of data extraction from tables the best. Along with Camelot, it was established that Google Document AI is also an acceptable venue for obtaining tables of satisfactory accuracy.

Literature

[1] Dengel, A., Kieninger, T. (1998) A Paper-to-HTML Table Converting System. DAS98, Int'l Association for Pattern Recognition Workshop on Document Analysis Systems. Deutsches Forschungszentrum für Künstliche Intelligenz.

[2] Wangt, Y., Phillipst, I. T., Haralick, R. (2001). Automatic table ground truth generation and a background-analysis-based table structure extraction method. In Proceedings of Sixth International Conference on Document Analysis and Recognition, Seattle, WA, USA, 2001, pp. 528-532. IEEE.

[3] Gatos B., Danatsas, D., Pratikakis, I., Perantonis, S.J. (2005). Automatic table detection in document images. In Proceedings of the Third International Conference on Advances in Pattern Recognition - Volume Part I (Bath, UK) (ICAPR'05). Springer-Verlag.

[4] Hao, L., Gao, L., Yi, X., Tang, Z. (2016). A Table Detection Method for PDF Documents Based on Convolutional Neural Networks. 2016 12th IAPR Workshop on Document Analysis Systems (DAS). IEEE.

[5] Kasem, M., Abdallah, A., Berendeyev, A., Elkady, E., Abdalla, M., Mahmoud, M., Hamada, M., Nurseitov, D., Taj-Eddin, I. (2024). Deep Learning for Table Detection and Structure Recognition: A Survey. ACM Computing Surveys, 56(12). Association for Computing Machinery.

[6] LeCun, Y., Bengio, Y., Hinton, G. (2015). Deep Learning. *Nature,* 521(7553), pp. 436-444.

[7] Kara, E., Traquair, M., Simsek, M., Kantarci, B., Khan, S. (2020). Holistic Design for Deep Learning-Based Discovery of Tabular Structures in Datasheet Images. Engineering Applications of Artificial Intelligence, 90(C).

[8] Xu, Y., Li, M., Cui, L., Huang, S., Wei, F., Zhou, M. (2020). LayoutLM: Pre-training of Text and Layout for Document Image Understanding. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '20),* pp. 1192-1200. ACM.

[9] OpenCV. Introduction to OpenCV-Python Tutorials (2025). URL: https://docs.opencv.org/4.x/d0/de3/tutorial_py_intro.html.

[10] Singer-Vine, J., & The pdfplumber contributors (2025). Pdfplumber (Version 0.11.6). URL: https://github.com/jsvine/pdfplumber/.

[11] Nurminen, A. (2013). Algorithmic Extraction of Data in Tables in PDF Documents (Master's thesis). Tampere University of Technology.

[12] Mehta, V. & The Camelot contributors (2025). Camelot (Version 1.0.0). URL: https://github.com/camelot-dev/camelot/.

[13] Google (2020). Introducing Document AI platform, a unified console for document processing. URL: https://cloud.google.com/blog/products/ai-machine-learning/google-cloud-announces-document-ai-platform/.

[14] Rogge, N. (2022). layoutlmv3-finetuned-funsd. URL: https://huggingface.co/nielsr/layoutlmv3-finetuned/.
[15] Smith, R. (2007). An Overview of the Tesseract OCR Engine. ICDAR '07: Proceedings of the Ninth International Conference on Document Analysis and Recognition, pp. 629-633. IEEE Computer Society.
[16] Google (2025). Process documents with Form Parser. URL: https://cloud.google.com/document-ai/docs/form-parser/.
[17] Rupšys, J. (2021). Lentelių atpažinimas (Bachelor's thesis). Vilnius University.
[18] Smock, B., Pesala, R., Abraham, R. (2022). PubTables-1M: Towards Comprehensive Table Extraction from Unstructured Documents. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Microsoft.