LaTeX Rainbow: Open Source Document Layout Semantic Annotation Framework

Changxu Duan
Technische Universität Darmstadt
Germany
changxu.duan@tu-darmstadt.de

Sabine Bartsch
Technische Universität Darmstadt
Germany
sabine.bartsch@tu-darmstadt.de

Abstract

Document Layout Analysis technology is advancing rapidly thanks to the large amount of high-quality labeled data. However, existing datasets comprised of document collections have shortcomings: (1) the hierarchical structure of papers is lost because labelling is done in terms of pages rather than documents; (2) content that is not part of the author's text such as page headers etc. is not filtered out; (3) papers included in a dataset are not likely to be up-to-date, i.e. they are not necessarily the latest version of a paper. We propose LATEX Rainbow, an open source annotation framework that can automatically annotate any LaTeX source code. This tool extends existing annotation methods by taking into account the properties of different datasets. It can produce token-level semantic structure annotations and preserve the paper's reading order as well as extract the table of contents i.e. information about the article's structure. LATEX Rainbow enables anyone to extend their datasets with the latest documents. This framework also has the flexibility of modifiable parsing rules and the potential to improve performance through parallelization. The project is open sourced on Github[1].

1 Introduction

Document Layout Analysis (DLA) is a critical technique in the realm of Natural Language Processing (NLP) since it improves the way machines process and understand text documents (Cui, 2021). A DLA system identifies and assigns semantic labels to diverse elements in a document. These elements encompass text blocks, figures, mathematical formulas, tables, among others. Each element has unique properties and makes up the entire document content.

DLA methods, which are fundamentally data-driven, have undergone rapid development and refinement over the past few years (Wang et al., 2021). The emergence of new NLP models and powerful computational infrastructures has expanded the horizon of possibilities in DLA applications (Huang et al., 2022). Moreover, the increasing availability and abundance of data resources is a cornerstone of DLA system development. Large volumes of diverse datasets have fostered comprehensive learning, thereby enabling the models to identify and categorize document elements more accurately, and to even extract high-level logical relationships between elements in documents.

As the field continues to evolve, the number of such datasets is proliferating at a substantial rate. The distinguishing factor among these datasets lies in their specialized focus areas. For instance, some datasets prioritize token-level annotation, enabling detailed element identification in a text document. Some datasets have particularly extensive document collections (Zhong et al., 2019). Others emphasize specific tasks such as the extraction of mathematical formulas, which is crucial for academic and research-related texts (Schmitt-Koopmann et al., 2022; Anitei et al., 2023). Additionally, certain datasets cater to providing the reading order, thereby enhancing document comprehension and navigation (Wang et al., 2021). However, there exist gaps in the current DLA datasets.

One of the key issues is that current annotations are made on a per page basis. There is no dataset that explicitly labels whether a element spanning two pages belongs to the same entity, e.g. a paragraph or itemized list. This means that an element spanning across multiple pages might be interpreted as two distinct entities instead of one continuous element. The coherence and context of the information could be compromised, which complicates the process of document interpretation.

Another problem is that elements that are irrelevant to the main content of the paper, such as watermarks, headers and page numbers, are not filtered out. Their inclusion often leads to noise in

[1] https://github.com/InsightsNet/texannotate

the data.

In addition to this the number of scientific publications has been increasing year by year and is showing no signs of slowing down. This means that current datasets are unlikely to incorporate the most recent papers. Moreover, authors of such resources do not always publish the code used in compiling the dataset, which hampers reproducibility of the process of dataset building as well as impeding scalability.

In this paper, we inherit several approaches for automatic PDF annotation of datasets and introduce a generalized framework that yields document-oriented, fine-grained, reading-ordered annotations that exclude extraneous content; the annotation builds on LaTeX source code directly. The output of our framework is one CSV table per PDF document and can be modified easily to suit the user's requirements. Our framework improves the accuracy and robustness of document information extraction, which we believe helps drive more accurate document comprehension in DLA systems. A more accurate and efficient document information extraction is also beneficial in areas such as automatic document summarization or information retrieval. This project is free software and available under the Apache 2.0 license.

2 Related Work

2.1 DLA Datasets

DLA datasets have their own characteristics. PubLayNet (Zhong et al., 2019) and DocLayNet (Pfitzmann et al., 2022) obtained particularly large amounts of labeling using automated and manual methods, respectively. XFUND (Xu et al., 2022) and TableBank (Li et al., 2020a) specialize in table extraction. FormulaNet (Schmitt-Koopmann et al., 2022) and IBEM (Anitei et al., 2023) focus on mathematical formulas, especially in-line formulas, which can easily be confused with plain-texts. DocBank (Li et al., 2020b) extended from TableBank, provides token-level labeling. ReadingBank (Wang et al., 2021) is extracted from Microsoft Word documents, which standardize the reading order of blocks within a page. M^{6}Doc (Cheng et al., 2023) extracted large-scale data using a half machine learning, half manual approach.

2.2 DLA Models

With the gradual enrichment of document layout data resources, model development is driven by increasingly larger datasets using different approaches. DocBank provides R-CNN models as a baseline. LayoutLM (Xu et al., 2020) and its variant VILA (Shen et al., 2022), make it possible to analyze layout from the 2D position of text. LayoutXLM (Xu et al., 2021b) and LiLT (Wang et al., 2022) implement multilingual layout recognition. LayoutLMv2 (Xu et al., 2021a) and DocFormer (Appalaraju et al., 2021) combine a language model and a visual coder to improve performance by learning not just the location of text, but also of non-textual visual features. LayoutLMv2 and LayoutLMv3 (Huang et al., 2022) use question answering models to extract high level semantic logical relationships in documents. DiT (Li et al., 2022) combines multiple vision modeling structures. Donut (Kim et al., 2022), on the other hand, changes the structure of model to a separate visual encoder and language model decoder without obtaining textual features directly from the document.

3 Methodology

Our REDACTED framework is built around three primary modules: the PDF compilation service, the LaTeX annotator, and the PDF extraction tool.

Figure 1 is a simplified representation of the labeling process. The implementation details of each module are explained in the next few sections.

3.1 PDF Compilation Service

Publications that accept submissions in the LaTeX format, including arXiv, often recommend using pdftex as their preferred rendering engine. This engine is integrated into the contemporary LaTeX distribution, TeXLive. Given the complexities in setting up this distribution, we opted for the Docker image of TeXLive 2023 to establish our compilation environment.

Automated LaTeX compilation presents a challenge, especially in pinpointing the master source file. This is because LaTeX allows for multiple `.tex` source files to be consolidated and compiled into one overarching master PDF. To navigate this challenge, we integrated arXiv's AutoTeX[2] automatic compilation system. AutoTeX, a Perl-based toolkit, excels at discerning the primary source file within a project. Our PDF compilation mechanism derives some of its functionalities from an open-source AutoTeX wrapper[3].

[2]https://metacpan.org/pod/TeX::AutoTeX
[3]https://github.com/andrewhead/texcompile

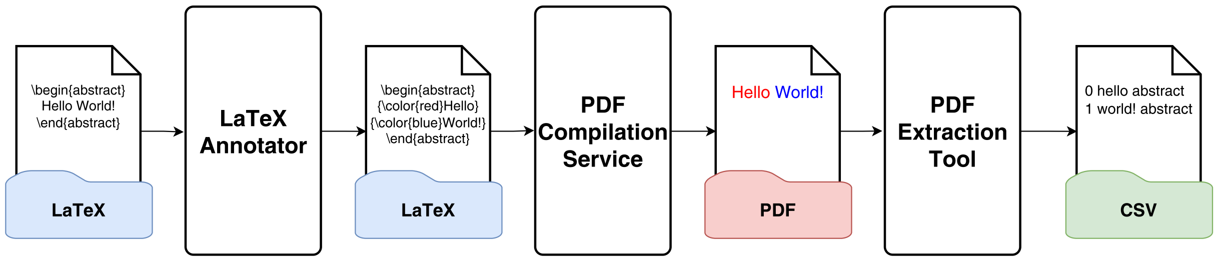

Figure 1: Labeling Process. (1) Initially, LaTeX source code is compiled, extracting the inherent color of every page, from the text color to the background color of rectangles on the page. This color information is pivotal as the process of annotating the LaTeX to PDF conversion primarily hinges on color alterations. Once compiled, the files undergo parsing and color-coding. Body text is highlighted with distinct colors, while text that has not been authored by the user remains unchanged. (2) Additionally, figures within the document are encased within borderless rectangles that are filled with a unique background hue. Every designated color corresponds to a specific segment of the LaTeX source code, ensuring traceability. After these annotations are made, the source code undergoes a second round of compilation into a PDF. Within this format, annotations are cross-referenced and aligned. (3) To finalize the process, the framework exports the location details of every text token and figure from the PDF, alongside the hierarchical relationship mapping between the annotations and the overall document.

However, during our practice, we observed AutoTeX's compilation regulations as overly stringent. There were instances when it halted the compilation due to minor errors, even when the same content had successfully passed arXiv's publication standards. To mitigate this, we refined AutoTeX's original code, allowing it to overlook certain errors and continue the PDF generation.

In conclusion, we combine TeXLive 2023, AutoTeX, and a Python-based API service into a single container. This container, accessible via HTTP, accepts source code and efficiently returns the compilation outcomes.

3.2 LaTeX Parsing and Annotation

This module processes the source code to assign color labels. Initially, it identifies the name of the main file based on the output from the PDF compilation service. For instance, if the service returns a file named `main.pdf`, the corresponding entry source file is `main.tex`.

To initiate the process, two specific lines of code are introduced at the start of the source file. The line `\pdfoutput=1` instructs the compiler to produce a PDF instead of Postscript, an alternative electronic publication format. On the other hand, `\interactionmode=1` signals the compiler to persist with the output generation, even if it encounters an error on a page.

Subsequently, we employ the Python package `pylatexenc`[4] to methodically traverse and parse the LaTeX source code, character by character. We've enhanced the system by integrating and extending the parsing rules from `pylatexenc` to ensure greater compatibility.

[4]https://github.com/phfaist/pylatexenc

Broadly, the source code comprises four segments: macro, environment, body, and comments. A macro in LaTeX begins with a backslash, and its arguments are encompassed within curly braces. We focus on labeling certain macros such as `\title{}` and `\author{}`, attributing their parameter literals with relevant semantic structure labels. The environment segment consists of entities encapsulated between starting and ending markers. For example, inside the `\begin{figure} ... \end{figure}`. Within this segment, elements are recursively analyzed.

Every identified text segment undergoes tokenization using the Spacy[5] tokenizer. For every recognized token, it gets substituted with `{\color[RGB]{0,0,1}<TOKEN>}`, while each identified figure is replaced with code `\colorbox[rgb]{0,0,0.1}{<FIGURE>}`.

The color **RGB** that can be assigned to the token ranges from #000000 to #fffff and it totals 16777216, the **rbg** that can be assigned to the figures ranges from (0,0,0) to (1,1,1) in steps of 0.1 and it totals 1331. Distinct colors are allocated to each segment. See Appendix A for detail.

We also insert rules that ensure any rectangle placed beneath an image does not disrupt the document's original layout. Concurrently, the article's hierarchical structure is captured and preserved within a tree data structure.

[5]https://spacy.io/

3.3 PDF Extraction and Annotation Export

This module is designed to extract page elements from the PDF file. It begins by identifying the position of each rectangle and then matches it to the respective figure element in the annotated source code by referencing its fill color. Following this, the module determines the color and position of each letter, utilizing the letter's color to match the corresponding annotation. Details of the extraction of colors are described in Appendix A.

During the export phase, the outputs are presented in a DataFrame. The initial part of the export showcases the table of contents' tree, which was created during the annotation process. This structured approach ensures that the succeeding page elements to be exported can be systematically assigned to their respective nodes. The DataFrame is finally saved as a CSV file.

4 Capabilities Overview

The tool generates a CSV file for every input LaTeX source project. This CSV file has two sections:

1. The initial rows represent the Table of Contents nodes. Here, each line denotes a tree node, with every node possessing a unique ID and an ID indicating its parent.

2. Every subsequent row denotes either a figure or token extracted from the PDF. These tokens are allocated a number indicating their reading order and a section ID, if they are part of the author's main content, starting from 0. A value of -1 in reading order indicates elements not penned by the author, and autogenerated by the LaTeX template. This facilitates a straightforward exclusion of such elements during further analysis. The label column encompasses semantic structure labels including: Abstract, Author, Caption, Equation, Figure, Footer, List, Paragraph, Reference, Section, Table, and Title.

5 Known Issues and Future Work

For extensive projects, consider the example of *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, which comprises 912 papers. The cumulative number of tokens in such projects might surpass the maximum number of colors that can be allocated. In these cases, the REDACTED framework is unable to perform the annotation. Our next steps include refining the color system to enhance its usability and distinctiveness.

Different publisher templates interpret LaTeX terms uniquely, making it challenging for our database to account for every variation. Thus, we heavily depend on the open-source community to contribute new rules. We also aim to glean insights from the functionalities of contemporary LaTeX IDEs such as Overleaf[6] or LaTeX-workshop[7].

In the future, our primary goal is to expand the REDACTED framework with parallelization capabilities. Given the containerized nature of our PDF compilation services, this transition should be seamless. We also plan to delve deeper into extracting layout semantics, such as pinpointing table headers and discerning logical correspondences.

6 Conclusion

LaTeX, a typesetting system commonly used for the publication of scientific documents, represents a highly valuable resource for Document Layout Analysis (DLA). This value arises primarily from the unique structure and nature of LaTeX source code. Unlike other document formats, LaTeX files include explicit markup that describes the structure and formatting of the document, which, in a sense, encapsulates the author's intent in writing.

Scientific publications inherently maintain a hierarchical and semantic structure, such as sections, subsections, figures, tables, equations, etc. These elements are all clearly labeled and defined within the LaTeX source code. This makes it considerably simpler for a DLA system to identify and categorize the diverse elements within a document, as they are already explicitly delineated and classified by the author's markup. Moreover, the author's intent can be inferred more effectively from the structural and semantic cues within the LaTeX code, leading to a more accurate and context-aware interpretation of the document.

Our framework is more than just yet another toolkit to the growing list of DLA resources. By ensuring versatility and adaptability as well as scalability, we aim for it to become a universal tool that can facilitate enhanced document analysis across multiple disciplines and applications. We sincerely hope that the open source community can derive many innovative uses and benefits from our solution.

[6]https://github.com/overleaf/overleaf
[7]https://github.com/James-Yu/LaTeX-Workshop

7 Acknowledgments

The framework reported is this paper was conducted within the research project InsightsNet (https://insightsnet.org/) which is funded by the Federal Ministry of Education and Research (BMBF) under grant no. 01UG2130A.

References

Dan Anitei, Joan Andreu Sánchez, José Miguel Benedí, and Ernesto Noya. 2023. The ibem dataset: A large printed scientific image dataset for indexing and searching mathematical expressions. *Pattern Recognition Letters*, 172:29–36.

Srikar Appalaraju, Bhavan Jasani, Bhargava Urala Kota, Yusheng Xie, and R. Manmatha. 2021. Docformer: End-to-end transformer for document understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 993–1003.

Hiuyi Cheng, Peirong Zhang, Sihang Wu, Jiaxin Zhang, Qiyuan Zhu, Zecheng Xie, Jing Li, Kai Ding, and Lianwen Jin. 2023. M6doc: A large-scale multi-format, multi-type, multi-layout, multi-language, multi-annotation category dataset for modern document layout analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15138–15147.

Lei Cui. 2021. *Document AI: Benchmarks, Models and Applications (Presentation@ICDAR 2021)*. DIL workshop in ICDAR 2021.

Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. 2022. Layoutlmv3: Pre-training for document ai with unified text and image masking. In *Proceedings of the 30th ACM International Conference on Multimedia*, MM '22, page 4083–4091, New York, NY, USA. Association for Computing Machinery.

Geewook Kim, Teakgyu Hong, Moonbin Yim, JeongYeon Nam, Jinyoung Park, Jinyeong Yim, Wonseok Hwang, Sangdoo Yun, Dongyoon Han, and Seunghyun Park. 2022. Ocr-free document understanding transformer. In *European Conference on Computer Vision (ECCV)*.

Junlong Li, Yiheng Xu, Tengchao Lv, Lei Cui, Cha Zhang, and Furu Wei. 2022. Dit: Self-supervised pre-training for document image transformer. In *Proceedings of the 30th ACM International Conference on Multimedia*, MM '22, page 3530–3539, New York, NY, USA. Association for Computing Machinery.

Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, Ming Zhou, and Zhoujun Li. 2020a. TableBank: Table benchmark for image-based table detection and recognition. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 1918–1925, Marseille, France. European Language Resources Association.

Minghao Li, Yiheng Xu, Lei Cui, Shaohan Huang, Furu Wei, Zhoujun Li, and Ming Zhou. 2020b. DocBank: A benchmark dataset for document layout analysis. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 949–960, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Birgit Pfitzmann, Christoph Auer, Michele Dolfi, Ahmed S. Nassar, and Peter Staar. 2022. Doclaynet: A large human-annotated dataset for document-layout segmentation. page 3743–3751.

Felix M. Schmitt-Koopmann, Elaine M. Huang, Hans-Peter Hutter, Thilo Stadelmann, and Alireza Darvishy. 2022. Formulanet: A benchmark dataset for mathematical formula detection. *IEEE Access*, 10:91588–91596.

Zejiang Shen, Kyle Lo, Lucy Lu Wang, Bailey Kuehl, Daniel S. Weld, and Doug Downey. 2022. VILA: Improving structured content extraction from scientific PDFs using visual layout groups. *Transactions of the Association for Computational Linguistics*, 10:376–392.

Jiapeng Wang, Lianwen Jin, and Kai Ding. 2022. LiLT: A simple yet effective language-independent layout transformer for structured document understanding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7747–7757, Dublin, Ireland. Association for Computational Linguistics.

Zilong Wang, Yiheng Xu, Lei Cui, Jingbo Shang, and Furu Wei. 2021. LayoutReader: Pre-training of text and layout for reading order detection. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4735–4744, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yang Xu, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, Wanxiang Che, Min Zhang, and Lidong Zhou. 2021a. LayoutLMv2: Multi-modal pre-training for visually-rich document understanding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2579–2591, Online. Association for Computational Linguistics.

Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. 2020. Layoutlm: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, KDD '20, page 1192–1200, New York, NY, USA. Association for Computing Machinery.

Yiheng Xu, Tengchao Lv, Lei Cui, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, and Furu Wei. 2021b. Layoutxlm: Multimodal pre-training for multilingual visually-rich document understanding.

Yiheng Xu, Tengchao Lv, Lei Cui, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, and Furu Wei. 2022. XFUND: A benchmark dataset for multilingual visually rich form understanding. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3214–3224, Dublin, Ireland. Association for Computational Linguistics.

Xu Zhong, Jianbin Tang, and Antonio Jimeno Yepes. 2019. Publaynet: largest dataset ever for document layout analysis. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1015–1022. IEEE.

A Annotation and Extraction of Colors Between LaTeX and PDF

For image annotations, we use the `\color[rgb]{}` format in LaTeX, spanning from (0, 0, 0) to (1, 1, 1) in incremental steps of 0.1, which has 1331 colors in the space. For token annotations, we use the `\color[RGB]{}` format, ranging from (0, 0, 0) to (255, 255, 255) with incremental steps of 1, yielding a total of 16777216 colors.

The task of extracting colors from PDFs is efficiently handled by the `pdfplumber`[8] package. This tool is proficient in pinpointing the position, font, and color of every character on a page. Additionally, it can determine the position and color attributes of all rectangles on the page, which encompasses both border and fill colors.

By default, `pdfplumber` utilizes the DeviceRGB color space, resulting in color extractions as tuples comprising three floating-point numbers. However, modern computer languages, including Python, sometimes struggle with accurately storing floating-point numbers. This inherent inaccuracy implies that color matching based on these numbers might be prone to errors, stemming from cumulative inaccuracies.

Our tests confirmed that all 1331 colors within image annotations' range were correctly matched. When `pdfplumber` extracts colors from token annotations' range, each tuple value is incremented in steps of 0.00392, for instance, (0, 1, 2) translates to extracted values of (0, 0.00392, 0.00784). Given that $1 \div 256 = 0.00390625$, we are already dealing with a discrepancy. To mitigate this, we employed matplotlib's `to_hex()`[9] method to ensure precise RGB value matches.

We also experimented with the `PyMuPDF`[10] package for color extraction, enticed by its capacity to extract colors as integer values. However, it uses the sRGB color space, which introduced mismatches between our annotations and extracted values. A notable misalignment was observed with colors (0, 0, 0) and (0, 1, 0) both being misconstrued as the singular color #000000 in sRGB. We will continue to study the impact of color for layout analysis and adjust the implementation described in this paper.

[8] https://github.com/jsvine/pdfplumber

[9] https://matplotlib.org/stable/api/_as_gen/matplotlib.colors.to_hex.html

[10] https://github.com/pymupdf/PyMuPDF