eSciBench: An Extensible Scientific PDF Extraction Benchmark

Noah Tremblay Taillon and Philippe Langlais
RALI/DIRO
Université de Montréal
noah.tremblay.taillon@umontreal.ca, felipe@iro.umontreal.ca

Abstract

Automatically extracting information from PDF documents (such as authors, affiliations, references, tables, equations) may be transformative in Digital Humanities where meta-data accompanying a document is typically manually collected, a cumbersome process. In this work, we conduct a systematic benchmarking of PDF extractors on a set of 100 scientific articles (1 949 pages) of the STEM domain that have been processed automatically, then carefully curated. Our benchmark, named *eSciBench* is openly accessible. Putting to the test 13 extractors on it reveals that although some extractors perform well overall, extracting information from scientific articles is far from a solved problem.

Keywords: PDF Extractors, Benchmark, STEM articles, arXiv

1. Introduction

Information extraction from PDF documents is crucial for discoverability because it transforms unstructured or semi-structured content into searchable, organized information. By automatically identifying and capturing key attributes—such as titles, authors, dates, topics, and keywords—metadata extraction enables efficient indexing and retrieval of digital resources.

We present *eSciBench*, an extensible benchmark designed for evaluating information extraction from scientific PDF articles. Currently, *eSciBench* encompasses 100 articles (1 949 pages) collected from arXiv (arXiv, 1991) with an annotation of 15 information types (metadata as well as specific information, such as text equations and tables) that have been manually curated. We believe *eSciBench* to be of high quality and well suited to evaluate extraction technology, and we actually put to test 13 open-source PDF extractors on 4 extraction tasks and discuss the performance obtained.

eSciBench is extensible (provided we have access to a LaTeX source document) and is freely available[1] (data, annotations as well as a carefully written tool suite developed for benchmarking technology), with the hope that it will foster research needed (as we will show) on extraction technology.

In Section 2 we discuss benchmarks dedicated to evaluate PDF extraction technology. We then detail in Section 3 issues with the otherwise well thought benchmark described by (Meuschke et al., 2023) and present our benchmark in Section 4. We report on benchmarking extractors in Section 5 and conclude in Section 6.

[1] https://github.com/TNoahT/eSciBench

2. PDF extraction benchmarks

Information extraction from PDF documents encompasses various tasks including layout analysis, optical character recognition as well as information specific extraction such as formulas, tables or bibliographical references. This diversity explains the numerous benchmarks that have been proposed so far.

Benchmarks such as *PubLayNet* (Zhong et al., 2019), *DocBank* (Li et al., 2020b) or *DocLayNet* (Pfitzmann et al., 2022) specialize in document layout analysis. Others, such as *Mined-PDF-Benchmark*[2] or *Extractous Benchmarks*[3] are dedicated to evaluating text extraction from PDF documents.

Many benchmarks target specific extraction needs. For instance *DocILE* (Šimsa et al., 2023) is dedicated to information extraction in business documents. *PubTabNet* (Zhong et al., 2020) and *TableBank* (Li et al., 2020a) are designed for evaluating table recognition.

Some benchmarks are evaluating several extraction scenarios, including information extraction on scientific articles. For instance, Ouyang et al. (2025) recently proposed *OmniDocBench*, a multi-level evaluation benchmark targeting both layout and content recognition. Scientific articles is only one of the 9 types of documents considered, leaving only 129 pages — not documents — in this domain. Further, they did not annotate scientific-specific labels such as *authors*, *keywords*, *affiliations*, and *emails*. Adhikari and Agarwal (2025) built on *DocLayNet* and gather a corpus of 6 document types, among which scientific articles. Here again, despite the large number of annotated pages, the

[2] https://github.com/circlestarzero/Miner-PDF-Benchmark?tab=readme-ov-file

[3] https://github.com/yobix-ai/extractous-benchmarks

Proceedings of the Fifteenth Language Resources and Evaluation Conference (LREC 2026), pages 7568–7580
11–16 May 2026.

scientific portion of it is small, and many scientific-specific annotations are missing (such as *authors*, *abstract*, *affiliations* and *references*).

Also, Meuschke et al. (2023) introduce a benchmark based on *DocBank* dedicated to scientific articles. We initially considered using this large benchmark but realized a number of critical issues with it (detailed in Section 3) that decided us to restart anew with *eSciBench*.

eSciBench builds on (Meuschke et al., 2023) and gathers scientific articles from arXiv without relying however on *DocBank* markup, but on LaTeX source files that we use to generate PDF documents (thus cleared of problems such as watermarking or layout inconsistencies due to compiler differences) and their layout and metadata annotations which we manually curated (see Section 4).

3. Meuschke's Benchmark

Among all the benchmarks listed above, the one of Meuschke et al. (2023) is likely the more suited to evaluate the meta-data and content extraction from scientific publications. Unfortunately, this benchmarks comes with implementation and methodological issues, raising concerns about the interpretability, fairness and reproducibility of the reported results.

3.1. Code Issues

The codebase used in (Meuschke et al., 2023) shows high coupling and inconsistent code reuse without documentation. For example, *load_data()*, the method which loads the ground-truth data, has been written originally for the benchmarking of the **Tabula** extractor (see Section 5.1 for a description of the extractors we benchmark). For most benchmarks, this function is used directly, but for others, the code is being duplicated with slight changes, as when benchmarking **PdfAct**, which code in turn is reused when benchmarking the **Science Parse** extractor. Similarly, the **Refextract**'s evaluation pipeline depends on **GROBID**'s code, which in turns calls the **CERMINE**'s XML parser. This set of dependencies in the evaluation code highly reduces clarity, hinders extension, and complicates validation of tool-specific logic.

3.2. Methodological Limitations

More critically, Meuschke et al. (2023)'s benchmark relies on *DocBank* as its ground truth, which introduces some issues.

First, *DocBank* was created using **Pdfplumber**[4], a tool known to struggle with complex layouts like tables in scientific articles (Adhikari and Agarwal, 2025); thus introducing potential errors in the ground truth, including cases documented in prior analyses of *DocBank* (Naik et al., 2022, p. 11). Further, **Pdfplumber** is inspired by the **Tabula**'s table extractor, thus biasing the reference in favour of this extractor.

Second, several important metadata elements, such as *keywords*, *emails*, *affiliations* and *publication dates*, are not asigned dedicated labels in *DocBank*. Instead, the corresponding tokens[5] are typically grouped under the *paragraph* label. As a result, tools that correctly extract these elements as structured metadata may be penalized during the evaluation process, since the ground-truth annotation treats them as regular paragraph texts.

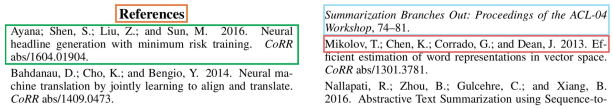

Figure 1: Two-column PDF layout of a reference list. Coloured boxes correspond to the *DocBank* markup shown in Figure 2.

Third, adjacent tokens with the same label are frequently merged into larger blocks, making it difficult to distinguish between consecutive references, or individual author names. This behaviour is partly explained by the fact that *DocBank* annotations are derived from PDF parsing rather than from the original LaTeX sources, as in *eSciBench*. While such segmentation artefacts are expected in PDF-derived datasets, they introduce ambiguities when the dataset is used as a fine-grained ground truth for extraction benchmarking.

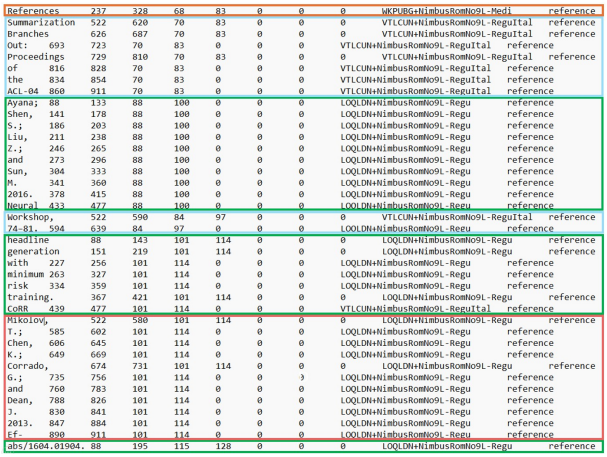

Figure 2: *DocBank* markup for the PDF extract illustrated in Figure 1. Texts of bounding boxes are listed in an order that prevents an easy gathering of a gold standard list of bibliographical entries.

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

[5] Space-separated sequences of characters.

```
...
"section": [
    {
      "number":"",
      "title":"References"
    }
],
"bibliography": [
    {
      "key": "ayana2016",
      "text": "Ayana; Shen, S.; Liu, Z.; and Sun, M. 2016.
         Neural headline generation with minimum risk training.
          CoRR abs/1604.01904."
    },
    {
      "key": "cho-translation",
      "text": "Bahdanau, D.; Cho, K.; and Bengio, Y. 2014.
         Neural machine translation by jointly learning to
         align and translate. CoRR abs/1409.0473."
    },
    ...
    {
      "key": "rouge",
      "text": "Lin, C.-Y. 2004. Rouge: A package for automatic
         evaluation of summaries. In Marie-Francine Moens, S. S
         ., ed., Text Summarization Branches Out: Proceedings
         of the ACL-04 Workshop, 74-81."
    },
    {
      "key": "word2vec",
      "text": "Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J.
          2013. Efficient estimation of word representations in
          vector space. CoRR abs/1301.3781."
    },
    ...
]
```

Figure 2 illustrates an especially pathological (but frequent) case where adjacent reference blocs (all marked by the *reference* label) are belonging to different bibliographical entries (see Figure 1 for the PDF layout). Tracking bounding boxes in the *DocBank* markup could potentially help to recover a meaningful gold annotation, but we found this to be a highly hazardous process.

Figure 3: Excerpt of *eSciBench*'s JSON gold standard for part of the layout illustrated in Figure 1.

In comparison, Figure 3 depicts *eSciBench*'s gold standard for this part of the layout. Clearly, the LaTeX and bibliographical material allows for a clearer separation of bibliographical entries than what can be reconstructed from PDF parsing alone.

Last but not least, it is important to note that only a few pages per PDF files are annotated in *DocBank*, while most extraction tools extract information from the whole document, without requiring pages to be separated first (which actually often perturbs extractors). In practice, this means that a page-level alignment of the output of each tool should be performed for a fair evaluation, which proved difficult to implement consistently across tools.

Meuschke et al. (2023)'s benchmark does not fully adress this issue, as page alignment is sometimes missing, or is inconsistently applied. For example, the benchmarking of **Refextract** uses token-based alignment via reference numbers, but no strategy is defined for unnumbered references or multi-page wrapping, which can greatly affect the tool's performance (not even mentioning frequent crashing of the evaluation tool in such cases).

In summary, while we believe *DocBank* to be useful for layout classification, its use as a fine-grained ground truth in PDF extraction benchmarking introduces strong biases up to the point that – we

believe – it undermines the validity of the evaluation conducted in (Meuschke et al., 2023).

4. *eSciBench*

4.1. Data

eSciBench is composed of 100 scientific documents, for a total of 1 949 pages, all sourced from arXiv over the 2009-2025 period and encompassing all arXiv categories, ensuring a variety of layouts, contents and bibliographical styles. Only articles with available LaTeX source files were selected, allowing accurate ground truth and PDF generation. See Table 1 for the distribution of articles in the arXiv categories and their volumetry.

Table 1: Number of articles and pages per category in the *eSciBench* dataset. EESS stands for Electrical Engineering and Systems Science.

Category	#PDFs	#pages
Physics	17	428
Mathematics	16	347
Computer Science	17	213
Quantitative Biology	16	218
Quatitative Finance	7	230
Statistics	2	30
EESS	18	174
Economics	8	309
Total	100	1 949

Our methodology for gathering annotation from the LaTeX source is inspired by Bast and Korzen (2017). We use pylatexenc's LatexNodes2Text class[6] to remove comments and turn the LaTeX content into a structured representation from which we extracted 15 labels (see Table 2).

As mentioned by Bast and Korzen (2017), extracting ground-truth data from a LaTeX file comes with some problems such as information included in different files, a variety of text commands, especially for equations, and the misuse of common macros. We found necessary to conduct manual intervention in the source code as well as validating some labels for ensuring annotation quality. We estimate that roughly 15 minutes on average were required per article, in great part for taking into account the variability with which equations can be entered in LaTeX. Details of all manipulations of the LaTeX structure and manual verifications conducted are provided in the *eSciBench* Github companion page. The resulting annotation which is streamed in JSON format is according to our standards of high quality. See Figure 3 for an example.

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

Table 2: Amount of individual ground-truth strings and blank-separated tokens for each label.

Label	# of strings	# of tokens
Abstract	101	15 744
Affiliation	404	4 258
Author	333	745
Caption	877	26 774
Email	190	193
Equation	3 321	59 984
Footer	344	7 667
Header	170	1 042
Keyword	250	493
List	244	34 370
Pub Date	29	82
Reference	3 818	88 538
Section	1 527	4 872
Table	253	22 139
Title	101	932

4.2. Evaluation and Metrics

The evaluation methodology we deployed is adapted from the foundation established by (Meuschke et al., 2023). The comparison of the extracted information with the gold standard one is conceptually simple. Yet, it encompasses in practice several normalization steps, mainly extractor specific, since the output format of extractors typically varies from one another. This includes lowering all the extracted material, removing new lines and extra spaces, normalizing specific characters such as quotation marks or apostrophes, removing the pipe symbols (|) often used by extractors to divide cell's content in tables or removing some information that we purposely removed from the gold standard annotation, such as reference numbers (e.g. [1]) in bibliographical references and table or figure numbering. All those steps (detailed in the companion Github) are done in great fairness and coded with transparency.

A similarity matrix collecting the edit-distance between each of the n extracted strings and the m ground-truth strings for each label, is computed resulting in a $n \times m$ matrix[7]. We apply the Hungarian algorithm[8] to this similarity matrix in order to find the best global one-to-one assignment (see Table 3) from which we compute the number of true positive (tp) pairs that we define as the extracted strings that have a similarity with a gold-standard string greater than 0.7[9]. Extracted strings that do not meet this

[7] We also implemented a *longest common substring* metric, with very similar results but a larger time response.

[8] We used the SciPy implementation (Virtanen et al., 2020).

[9] The threshold of 0.7 is following the recommenda-

Table 3: Illustration of a similarity matrix. Bold figures indicate the one-to-one assignment computed by the Hungarian algorithm, squared cells identify true positives (similarity over 0.7), underlined and overlined strings are false positives and negatives respectively.

Extracted ↓ Gold →	onel luis alcaraz lópez	andy goldschmidt	a. sikov	shan zuo
alcaraz lópez	**0.722**	0.207	0.190	0.286
andy goldschmidt	0.256	**1.000**	0.333	0.333
audio-visual separator	**0.356**	0.316	0.267	0.267

criterion are false positives (fp), while gold-standard strings not matched are false negatives (fn). With these numbers, we classically report precision (P=tp/tp+fp) and recall (R=tp/tp+fn) figures that we combine into an f-mesure (F_1=2 × P × R/P+R). Other figures are also reported by our evaluation script such as accuracy (Acc=tp/tp+fp+fn), BLEU (Papineni et al., 2002) and ROUGE-n (Lin, 2004), for $n \in \{1, 2, 3\}$.

4.3. Tasks

Following (Tkaczyk et al., 2015) and (Meuschke et al., 2023), extractors are evaluated along 4 tasks:

Metadata extraction consists in identifying *abstracts* (some articles may have several ones), *affiliations*, *authors*, *emails*, *keywords*, *publication date* and *title*. The different metadata labels we consider are inspired by the description of metadata for articles and journals from (Byers and Lujano, 2023). We preserve as much as possible the structure of an information. For instance, the gold annotation for *authors* is structured as shown here:

```
"author": [
{
  "author_name": ...,
  "affiliation": ...,
  "email": ...
},
...
]
```

General extraction consists in identifying *captions*, *equations*, *footers*, *headers* and *sectioning*. For instance, equations encountered in an article are serialized into strings as in:

```
"equation": [
  "G=e(X,Y)=(x_q_1,..., x_q_l),",
  "f_i^j = M_i:i+k_j -1 W_j",
  "x_t = 1/m S_i=1^m f_w_i.",
  ...
]
```

tions of (Meuschke et al., 2023).

Reference extraction consists in the identification of each bibliographical entry in the article. Note that a bibliographical reference is a string which is not parsed into its constituants such as authors, date of publication, title, tribune, etc. And this, even if some extractors are able to further parse reference strings such as **GROBID** (Lopez, 2009).[10]

Table extraction consists in extracting the text of tables in an article. Each table text in *eSciBench* is streamed into a single string, following the standard row order (first row, then second, etc.). For instance, the table illustrated in Figure 4 is streamed as:

```
Sentences Score S_A: Terry's career as a photographer came after he failed to make it as a punk rock musician. S_B^+: He got his first big break in 1994 with a shoot for Vibe magazine. 0.9885 S_B^-: The photographer has also directly music videos in his time. 0.5198
```

Table 1: Example outputs of neural coherence model.

Sentences	Score
S_A: Terry's career as a **photographer** came after he failed to make it as a punk rock musician.	
$S_B{}^+$: **He** got his first big break in 1994 with a **shoot** for Vibe magazine.	**0.9885**
$S_B{}^-$: The photographer has also directly music videos in his time.	0.5198

Figure 4: Part of a table found in (Wu and Hu, 2018).

5. Experiments

5.1. Extractors

We tested 13 extractors on *eSciBench*, all open-source tools, capable of full-text PDF extraction.

[10] https://github.com/kermitt2/grobid

Not all extractors are capable of extracting all information annotated in *eSciBench*: See Table 4 for their main characteristics.

Camelot [11]: A Python library for table extraction using the pypdfium2[12] library as the image conversion backend. Camelot supports four extraction modes: "Stream" using the text extraction tool PDFMiner[13] to parse tables with cells delimited by whitespaces, "Lattice" using the computer vision library OpenCV[14] to parse tables delimited by lines, "Network" using bounding boxes encoded in the PDF to locate the tables, and "Hybrid" combining the Network and Lattice parsers. The Network option is used for its superior results.

CERMINE (Tkaczyk et al., 2015): A Java-based tool using iText[15], a PDF manipulation library, for text extraction, the Support Vector Machines library LibSVM (Chang and Lin, 2011) and rules to parse the matadata, and conditional random fields (CRFs) based on the Gaussian Mixture Model from the MALLET toolkit (McCallum, 2002) for reference parsing. The JAR executable is used for this benchmark, and the JATS XML output file is parsed with Python's *xml.etree.ElementTree* API (Python Software Foundation, 2025) during the benchmarking.

Docling (Auer et al., 2024): A recent and popular open-source toolkit for Python using specialized transformer models trained to convert documents into a structured representation. We use the default settings and models. The layout analysis is conducted by an adaptation of RT-DETR (Zhao et al., 2024), re-trained on DocLayNet (Pfitzmann et al., 2022). The table detection uses the TableFormer model (Nassar et al., 2022). The extracted information is saved in a structured JSON file. Other layout models and OCR engines are available.

GROBID (Lopez, 2009): A machine learning software for scientific PDFs, using the PDF to XML command line executable pdfalto[16] for layout information. A cascade of models, including CRF and deep learning models, categorize the data into 68 final labels. The text classification framework DeLFT[17] complements the CRF models with the layout information. GROBID is executed locally using Docker with raw affiliations and citations enabled. The tool creates an TEI XML file, which is parsed with *xml.etree.ElementTree*.

Nougat (Blecher et al., 2023): A transformer-based model converting scientific documents into Markdown, using a Donut (Kim et al., 2022) architecture, the Swin Transformer (Liu et al., 2021) (a hierarchical vision transformer) for visual encoding, and the mBart decoder (Lewis et al., 2019). Nougat extracts Tables and equations in LaTeX format; pylatexenc's library is then used in the evaluation to turn the data to text. The default model is used for the evaluation.

PdfAct [18]: A rule-based extraction tool for scientific articles, initially called Icecite (Bast and Korzen, 2017). **PdfAct** uses the PDF manipulation library Apache PDFBox (The Apache Software Foundation, 2025) and outputs a JSON file. The JAR executable is used, with the "–include-roles <roles>" option, including all possible semantic roles.

Pdfplumber : A Python library for text and table extraction, using pdfminer to extract the text. It is used in this benchmark for table parsing with an optimal line detection strategy.

PyMuPDF [19]: A Python library extending MuPDF (Artifex Software, Inc, 2025) and supporting text and table extraction. We used the default "lines" strategy for its superior results, using a page's vector graphics to detect tables.

RefExtract [20]: A Python reference extraction library using the PDF converter pdftotext (Glyph & Cog, LLC, 2004) and outputting the structured bibliographical data in a JSON file. We compiled this library locally via Docker to access the latest update.

Science Beam Parser [21]: A GROBID-based parser using the pdf to XML converter pdfalto and deep learning models to extract data. Tesseract is used for OCR, and the output is a TEI XML file, parsed with *xml.etree.ElementTree* for the evaluation. We used the Dockerized version.

[11] https://github.com/camelot-dev/camelot/
[12] https://github.com/pypdfium2-team/pypdfium2
[13] https://github.com/pdfminer/pdfminer.six
[14] https://opencv.org
[15] https://github.com/itext/itext-java
[16] https://github.com/kermitt2/pdfalto
[17] https://github.com/kermitt2/delft
[18] https://github.com/ad-freiburg/pdfact
[19] https://github.com/pymupdf/PyMuPDF
[20] https://github.com/inspirehep/refextract
[21] https://gitlab.coko.foundation/sciencebeam/sciencebeam-parser

Table 4: Main characteristics of the extractors we tested. T, G, M and R stand respectively for Table- General- Metadata- and Reference-extraction tasks. The underlined output format is the one we used.

Tool	Version	Task	Technology	Output
Camelot	1.0.0	T	pdfium, PDFMiner, OpenCV	CSV, JSON, Excel, HTML, Markdown, Sqlite, Table object
CERMINE	1.13	G, M, R	iText, Docstrum, SVM, rules, CRF	JATS XML
Docling	2.76	G, M, T, R	transformer	HTML, Markdown, JSON,Text, Doctags, WebVTT
GROBID	0.8.2	G, M, R, T	CRF, deep learning, pdfalto	TEI XML
Nougat	0.1.0-small	G, M, T, R	Donut, transformer, mBart	Mathpix Markdown
PdfAct	2022-01-04	G, M, T, R	PDFBox, rules	TXT, XML, JSON
Pdfplumber	0.11.7	T	PDFMiner	CSV, JSON, TXT, PDF object
PyMuPDF	1.26.3	T	MuPDF, Tesseract, rules	Document object
Refextract	0.2.6	R	pdftotext	JSON
Science Beam	0.1.8	G, M, R	deep learning, pdfalto, Tesseract	TEI XML, JSON
Science Parse	3.0.1	G, M, R	deep learning	JSON
Tabula	2.10.0	T	PDFBox	CSV, DataFrame, JSON, TSV
Unstructured	0.18.10	G, M, T	deep learning, Tesseract	Element object

Science Parse [22]: An Ai2 tool forked parser using machine learning to extract data into structured JSON files. UCREL NLP Group's version adds back the Dockerfile, which we used to execute this tool.

Tabula [23]: A Java-based table extractor interfaced for Python. Similarly to **Camelot**, it uses Apache PDFBox and OpenCV to parse tables (Aristarán and Tigas, 2013). Tabula supports "Stream" and "Lattice" parsing methods; the first one was used for benchmarking.

Unstructured [24]: A modular tool using layout detection with the visual detection tool Detectron2[25] and Tesseract for OCR. The "hi-res" extraction strategy is used for optimal layout classification.

Note that some extractors are computationally more demanding than others. In particular[26], we found **Unstructured** to be by far the slowest tool (with an average of 134 seconds per PDF), **Nougat** being the second slowest (requiring 13 seconds) and **Science Parse** being the fastest (1.1 second).

[22] https://github.com/UCREL/science-parse

[23] https://github.com/chezou/tabula-py

[24] https://github.com/Unstructured-IO/unstructured

[25] https://github.com/facebookresearch/detectron2

[26] We used the same Linux workstation to benchmark all tools.

5.2. Results

The full set of metrics for all extractors and all labels are presented in Appendix. We observe that extractors perform very differently on specific labels.

For instance, **PdfAct** is reasonably good at extracting *Titles* but hardly extracts labels such as *Abstract* or *Authors*. We also observe important differences between the F_1 score we compute and the one we computed according to (Meuschke et al., 2023). This is due in great part to the high quality of our gold annotation, while the one in *DocBank* is problematic, as we discussed in Section 3. It is also explained by the document level evaluation, and the alignment procedure it encompasses, which is overlooked in (Meuschke et al., 2023).

For an easier comparison of label extraction performance, we report the F_1 score of extractors for the general extraction task (Table 5), the meta-data task (Table 6) and the reference and table tasks (Table 7). Several observations can be made.

We observe that overall, the general extraction task is difficult. For instance, **GROBID**, an overall good extractor, only receives a F_1 score of 51.2 for equations extraction. *Captions* are best extracted by **Docling**, but tend to be mixed up with regular

Table 5: F_1 (%) of the general extraction task. Bold figures are the best for a given label.

Label	**CE**	**D**	**G**	**N**	**PA**	**SB**	**SP**	**U**
Caption	-	**81.0**	62.2	-	77.6	-	-	78.6
Equation	-	**71.7**	51.2	67.4	0.0	39.7	-	40.7
Footer	-	27.0	**59.1**	39.1	20.7	-	-	7.1
Header	-	44.0	-	-	**52.2**	-	-	30.3
Section	54.4	79.5	60.8	**87.6**	39.9	57.4	81.7	87.3
List	-	2.2	-	**6.0**	-	-	-	3.3

paragraphs, explaining the average scores across extractors.

The fact that **Docling**, **Nougat** and **Unstructured** underperform when extracting *Lists* demonstrates the difficulty of this task. These last two extractors consider bibliographical references as list items, requiring heuristics to filter them. Also, **Docling** has issues decoding characters in certain documents, hindering its scores in all tasks. For example, the following was extracted as a section title:

```
"section":
    ☆ ■♡/a116/a114♢"✉"/a116†♢♡
```

Equation extraction is generally difficult for all tools, and the best F_1 score (71.7) is obtained by **Docling**, followed by **Nougat**. This overall difficulty can be caused by a difference in the extracted mathematical characters and the ground-truth ones. For example, $\sqrt{x}$ and $\sqrt{x}$ could be two ways of representing the square root symbol. Since **Nougat** and the ground truth both use pylatexenc, this might put this extractor at an advantage. This also means that better normalization should be carried out for this, which we leave for future work.

The *Footer* and *Header* low scores may be explained by some tools that include page numbers, while our ground-truth does not, as well as an overall difficulty of tools to differentiate regular text from footnotes when the police sizes are similar. Regarding the variability of results on *Section* extraction, we noticed that some tools have a difficulty to distinguish bold text from section title.

The metadata extraction task is the easiest to perform, and **GROBID** shines by outperforming other extractors on 5 out of the 7 metadata labels, and being close to the best performing tool (**Science Parse**) for labels *Affiliation* and *Title*. This said, some extractors such as **PdfAct** have problems with extracting metadata information while those labels are in theory taken care of according to their documentation.

The bad results of **Cermine** on *Pub Date* are due to the tool extracting only the year of publication, while scientific articles (and therefore our reference) also mention the day and the month, rendering the alignement with the ground-truth always unsuccessful. Also, **Science Parse** extracts the years from the encoding of the PDF, which is not always matching the publication date inside the article.

The variability of the placement for author's *affiliations* and *emails* in the PDFs seems to challenge all tools. For example, some articles present these informations under the title, while others can place them in a footnote or after the bibliography.

We observe that when extracting *References*, tools are overall good, and the task is best performed by **GROBID**, **Science Beam Parser**, and **Nougat**. **PdfAct** tends to miss some references, while over-segmenting others. It also struggles extracting references that are scattered on different columns or pages. As mentioned before, **Nougat** and **Unstructured** label references as list items, requiring heuristics to differenciate them.

Table 6: F_1 (%) of the reference and table extraction tasks.

System	*Reference*	*Table*
Camelot	-	14.6
Pdfplumber	-	23.7
PyMuPDF	-	19.1
Tabula	-	31.4
Unstructured	81.9	**78.5**
Docling	79.1	74.1
GROBID	**95.5**	42.9
Nougat	92.3	71.9
PdfAct	59.0	0.00
CERMINE	87.5	-
RefExtract	84.5	-
Science Beam Parser	92.8	-
Science Parse	85.2	-

For the *Table* extraction task, **Unstructured**, **Docling** and **Nougat**, are the best, while other extractors show strong limitations. **Camelot** has difficulty differentiating regular paragraphs on multiple columns from tables, which is greatly affecting it's score. On the other hand, **PyMuPDF** seems very careful in its extraction, only treating a small portion of available tables, hence low recall.

Overall, we observe that extractors using ma-

Table 7: F_1 (%) of the metadata extraction task. Bold figures are the best for a given label.

Label	CE	D	G	N	PA	SB	SP	U
Abstract	88.1	-	**94.9**	-	3.6	89.4	93.1	-
Affilliation	52.8	-	**76.3**	-	0.0	60.6	-	-
Author	83.8	-	92.2	-	0.0	84.5	**97.1**	-
Email	71.4	-	**71.6**	-	-	-	-	-
Keyword	-	-	**85.5**	-	0.0	72.2	-	-
Date	0.0	-	**84.6**	-	-	72.3	0.0	-
Title	92.8	88.1	97.5	84.9	89.1	88.9	**98.5**	90.1

chine learning dominate the charts, but rule-based **PdfAct** outperforms **Docling** and **Unstructured** two deep learning extractor on extracting *headers*.

We found **GROBID** to be the best tool overall, with the best performance, showing the best score for 7 out of 15 labels, and being second to best when extracting *author names* and *titles*. Surprisingly, **CERMINE**, a tool without updates since 2018, performs generally well.

6. Conclusion

We presented *eSciBench*, an extensible scientific PDF extraction benchmark composed of 100 LaTeX documents retrieved from arXiv (1 949 pages), parsed with a custom LaTeX parser and converted into a curated JSON representation that allows the evaluation of 4 extraction tasks (general, metadata, table and references).

Using a framework inspired by (Meuschke et al., 2023), we evaluated the performance of 13 open-source PDF extraction tools. **GROBID** comes out as the best overall extractor, with the best F_1 scores for *abstract*, *affiliation*, *email*, *keyword*, *publication date*, *footer* and *reference* extraction. **Science Parse** excels in *author name* and *title* extraction, **Docling** gets the best scores for *equation*, **Nougat** for *section* title and *list* items, and **Unstructured** is the leader for *caption* and *table* extraction. **PdfAct**, the only rule-based tool among the top performing extractors, has the best F_1 score for *header* extraction. Overall, *table*, *equation* and *header* extraction remain challenging across most tools.

The variability in top-performing tools across different extraction tasks supports the idea of a hybrid extraction pipeline, combining specialized tools for specific content types, to achieve the most accurate and comprehensive results.

Future works include considering a larger set of extractors, and investigating more systematically their parametrization. In particular, in the present study we only tested extractors that could run on a simple desk computer equipped with a 24Gb memory GPU. Extractors relying on larger models definitely deserve to be investigated. Also, limitations listed in the following section are worth being investigated.

7. Limitations

While we believe *eSciBench* to be one of a kind extraction benchmark, there are limitations that we would like to discuss.

The first limitation is that *eSciBench* does not encompass the evaluation of full-text extraction from PDF. This is due to the fact that we were more interested in markup that we know digital library professionals are striving to annotate manually or semi-automatically. Also, full-text extraction is arguably a more specific problem that requires dedicated metrics. In particular, text extractors vary in the way the text is being output, requiring a dedicated alignment algorithm that we thought was out of the scope of this work.

Second, table extraction in *eSciBench* is based on the LaTeX source, which does not necessarily match the natural reading order, putting at an advantage extractors that operate with the same order as the ground truth. We could have – but did not – introduce a metric sensible to this order (by introducing a specific alignment procedure).

Third, because we carefully curated gold annotations, *eSciBench* is currently small with 100 document and 1 949 pages, while larger datasets are available (although arguably less accurate) such as *DocBank* or PubLayNet. We believe that 100 carefully annotated articles is already good to diagnose extractors.[27] Moreover, the straightforward annotation process we sketched here, and which is further described in the companion Github allows *eSciBench* to be easily expanded.

Fourth, part of the information to extract is indeed structured (in particular bibliographical references and tables), but *eSciBench* is currently treating them as strings, loosing some useful information.

Finally, *eSciBench* is currently composed exclusively of natural science documents. Expending the

[27] Sampling half of the material leads to similar results overall.

dataset with humanities and human science publications would introduce more diverse document layouts, enabling broader evaluation of extraction tool performance.

Acknowledgments

We are thankful to the salient comments made by reviewers on the submitted version of this article.

8. Bibliographical References

Narayan S. Adhikari and Shradha Agarwal. 2025. A Comparative Study of PDF Parsing Tools Across Diverse Document Categories.

Manuel Aristarán and Mike Tigas. 2013. Introducing Tabula. https://source.opennews.org/articles/introducing-tabula/.

Artifex Software, Inc. 2025. MuPDF.

arXiv. 1991. arxiv: Open access to scholarly articles. Accessed: 2025-10-20.

Christoph Auer, Maksym Lysak, Ahmed Nassar, Michele Dolfi, Nikolaos Livathinos, Panos Vagenas, Cesar Berrospi Ramis, Matteo Omenetti, Fabian Lindlbauer, Kasper Dinkla, Lokesh Mishra, Yusik Kim, Shubham Gupta, Rafael Teixeira de Lima, Valery Weber, Lucas Morin, Ingmar Meijer, Viktor Kuropiatnyk, and Peter W. J. Staar. 2024. Docling technical report.

Hannah Bast and Claudius Korzen. 2017. A Benchmark and Evaluation for Text Extraction from PDF. In *2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pages 1–10, Toronto, ON, Canada. IEEE.

Lukas Blecher, Guillem Cucurull, Thomas Scialom, and Robert Stojnic. 2023. Nougat: Neural Optical Understanding for Academic Documents.

Andy Byers and Ivonne Lujano. 2023. Article and journal metadata - OA Journals Toolkit.

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3).

Glyph & Cog, LLC. 2004. Pdftotext(1). https://linux.die.net/man/1/pdftotext.

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)*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension.

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.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows.

Patrice Lopez. 2009. Grobid: Combining automatic bibliographic data recognition and term extraction for scholarly publications. In *Proceedings of the 13th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2009)*, volume 5714 of *Lecture Notes in Computer Science*, pages 473–474. Springer.

Andrew Kachites McCallum. 2002. MALLET: A Machine Learning for Language Toolkit.

Norman Meuschke, Apurva Jagdale, Timo Spinde, Jelena Mitrović, and Bela Gipp. 2023. A Benchmark of PDF Information Extraction Tools using a Multi-Task and Multi-Domain Evaluation Framework for Academic Documents. volume 13972, pages 383–405.

Shivam Naik, Khurram Azeem Hashmi, Alain Pagani, Marcus Liwicki, Didier Stricker, and Muhammad Zeshan Afzal. 2022. Investigating attention mechanism for page object detection in document images. 12:7486.

Ahmed Nassar, Nikolaos Livathinos, Maksym Lysak, and Peter Staar. 2022. TableFormer: Table Structure Understanding with Transformers.

Linke Ouyang, Yuan Qu, Hongbin Zhou, Jiawei Zhu, Rui Zhang, Qunshu Lin, Bin Wang, Zhiyuan Zhao, Man Jiang, Xiaomeng Zhao, Jin Shi, Fan

Wu, Pei Chu, Minghao Liu, Zhenxiang Li, Chao Xu, Bo Zhang, Botian Shi, Zhongying Tu, and Conghui He. 2025. OmniDocBench: Benchmarking Diverse PDF Document Parsing with Comprehensive Annotations.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Birgit Pfitzmann, Christoph Auer, Michele Dolfi, Ahmed S. Nassar, and Peter W. J. Staar. 2022. DocLayNet: A Large Human-Annotated Dataset for Document-Layout Analysis. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3743–3751.

Python Software Foundation. 2025. The ElementTree XML API.

Štěpán Šimsa, Milan Šulc, Michal Uřičář, Yash Patel, Ahmed Hamdi, Matěj Kocián, Matyáš Skalický, Jiří Matas, Antoine Doucet, Mickaël Coustaty, and Dimosthenis Karatzas. 2023. DocILE Benchmark for Document Information Localization and Extraction.

The Apache Software Foundation. 2025. Apache PDFBox | A Java PDF Library. https://pdfbox.apache.org/.

Dominika Tkaczyk, Pawel Szostek, Mateusz Fedoryszak, Piotr Jan Dendek, and Lukasz Bolikowski. 2015. CERMINE. *International Journal on Document Analysis and Recognition (IJDAR)*, 18(4):317–335.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272.

Yuxiang Wu and Baotian Hu. 2018. Learning to extract coherent summary via deep reinforcement learning. *CoRR*, abs/1804.07036.

Yian Zhao, Wenyu Lv, Shangliang Xu, Jinman Wei, Guanzhong Wang, Qingqing Dang, Yi Liu, and Jie Chen. 2024. DETRs Beat YOLOs on Real-time Object Detection.

Xu Zhong, Elaheh ShafieiBavani, and Antonio Jimeno Yepes. 2020. Image-based table recognition: Data, model, and evaluation. In *European Conference on Computer Vision*, pages 564–580, Glaskow, UK.

Xu Zhong, Jianbin Tang, and Antonio Jimeno Yepes. 2019. PubLayNet: Largest dataset ever for document layout analysis.

Table 8: Results grouped by extraction tool (1/2). Boldface indicates the best value for each content element type, if data has been extracted. A missing label for a tool indicates that this type of annotation is not supported. A score of 0.000 means that while the corresponding label is supported, no data has been extracted. **Lev** stands for the Levenshtein distance between the extracted and gold materials.

Tool	Label	tp	fp	fn	Acc	F_1	P	R	Lev
Camelot	Table	143	1,565	110	0.079	0.146	0.084	0.565	0.073
CERMINE	Abstract	85	8	15	0.787	0.881	0.914	0.850	0.777
	Affiliation	150	23	245	0.359	0.528	0.867	0.380	0.344
	Author	268	45	59	0.720	0.838	0.856	0.820	0.715
	Email	106	3	**82**	0.555	0.714	0.972	**0.564**	0.553
	Pub Date	0	28	29	0.000	0.000	0.000	0.000	0.000
	Reference	2,938	54	789	0.777	0.875	**0.982**	0.788	0.766
	Section	861	831	613	0.374	0.544	0.509	0.584	0.365
	Title	90	4	10	0.865	0.928	0.957	0.900	0.855
Docling	Caption	**701**	153	**176**	**0.681**	**0.810**	0.821	**0.799**	**0.648**
	Equation	2,428	1,025	893	**0.559**	**0.717**	**0.703**	0.731	0.437
	Footer	**281**	1,455	**63**	0.156	0.270	0.162	**0.817**	0.151
	Header	**157**	387	**13**	0.282	0.440	0.289	**0.924**	0.273
	List	26	2,130	218	0.011	0.022	0.012	0.107	0.008
	Reference	2,565	**9**	1,253	0.655	0.791	0.963	0.672	0.645
	Section	1,313	464	214	0.659	0.795	0.739	0.860	0.654
	Table	192	73	61	0.589	0.741	0.725	0.759	0.563
	Title	89	12	12	0.788	0.881	0.881	0.881	0.782
GROBID	Abstract	93	**2**	8	**0.903**	**0.949**	**0.979**	0.921	**0.891**
	Affiliation	**291**	69	**113**	**0.617**	**0.763**	0.808	**0.722**	**0.583**
	Author	**320**	41	**13**	0.856	0.922	0.886	**0.961**	0.846
	Caption	532	302	345	0.451	0.622	0.638	0.607	0.435
	Email	**107**	**2**	83	**0.557**	**0.716**	**0.982**	0.563	**0.554**
	Equation	1,806	1,921	1,515	0.345	0.512	0.485	0.544	0.280
	Footer	159	**35**	185	**0.420**	**0.591**	**0.820**	0.462	**0.413**
	Keyword	**221**	46	**29**	**0.747**	**0.855**	0.828	**0.884**	**0.742**
	Pub Date	**22**	**1**	**7**	**0.733**	**0.846**	**0.957**	**0.759**	**0.724**
	Reference	3,702	229	**116**	**0.915**	**0.955**	0.942	**0.970**	**0.906**
	Section	1,093	976	434	0.437	0.608	0.528	0.716	0.434
	Table	81	44	172	0.273	0.429	0.648	0.320	0.254
	Title	97	1	4	0.951	0.975	0.990	0.960	0.946
PdfAct	Abstract	17	820	84	0.018	0.036	0.020	0.168	0.018
	Affiliation	0	0	403	0.000	0.000	0.000	0.000	0.000
	Author	0	0	333	0.000	0.000	0.000	0.000	0.000
	Caption	624	**108**	253	0.634	0.776	**0.852**	0.712	0.591
	Equation	0	0	2,936	0.000	0.000	0.000	0.000	0.000
	Footer	174	1,169	163	0.116	0.207	0.130	0.516	0.111
	Header	142	**270**	17	**0.331**	**0.497**	**0.345**	0.893	**0.316**
	Keyword	0	0	247	0.000	0.000	0.000	0.000	0.000
	List	0	0	244	0.000	0.000	0.000	0.000	0.000
	Reference	1740	338	2,078	0.419	0.590	0.837	0.456	0.412
	Section	1101	2895	422	0.249	0.399	0.276	0.723	0.248
	Table	0	0	253	0.000	0.000	0.000	0.000	0.000
	Title	85	7	15	0.794	0.885	0.924	0.850	0.776

Table 9: Results grouped by extraction tool (2/2). Boldface indicates the best value for each content element type, if data has been extracted. A missing label for a tool indicates that this type of annotation is not supported. A score of 0.000 means that while the corresponding label is supported, no data has been extracted. **Lev** stands for the Levenshtein distance between the extracted and gold materials.

Tool	Label	tp	fp	fn	Acc	F_1	P	R	Lev
Nougat	Equation	**2,539**	1,674	**782**	0.508	0.674	0.603	**0.765**	**0.441**
	Footer	166	340	178	0.243	0.391	0.328	0.493	0.240
	List	36	917	208	0.031	0.060	0.038	0.148	0.025
	Reference	**3,496**	264	332	0.856	0.923	0.930	0.916	0.816
	Section	**1,375**	238	**152**	**0.779**	**0.876**	0.852	**0.900**	**0.771**
	Table	150	**14**	103	0.562	0.719	**0.915**	0.593	0.525
	Title	76	2	25	0.738	0.849	0.974	0.752	0.723
Pdfplumber	Table	78	326	175	0.135	0.237	0.193	0.308	0.297
PyMuPDF	Table	31	40	222	0.106	0.191	0.437	0.123	0.096
RefExtract	Reference	3,689	1,221	192	0.732	0.845	0.751	0.966	0.719
Science Beam	Abstract	84	3	17	0.808	0.894	0.966	0.832	0.788
	Affiliation	183	**17**	221	0.435	0.606	**0.915**	0.453	0.422
	Author	273	40	60	0.732	0.845	0.872	0.820	0.725
	Equation	1004	**738**	2,317	0.247	0.397	0.576	0.302	0.197
	Keyword	152	**19**	98	0.565	0.722	**0.889**	0.608	0.562
	Pub Date	17	**1**	12	0.567	0.723	0.944	0.586	0.558
	Reference	3,361	66	457	0.865	0.928	0.981	0.880	0.857
	Section	983	914	544	0.403	0.574	0.518	0.644	0.397
	Title	84	4	17	0.800	0.889	0.955	0.832	0.795
Science Parse	Abstract	**94**	7	**7**	0.870	0.931	0.931	**0.931**	0.860
	Author	**320**	**6**	**13**	**0.944**	**0.971**	**0.982**	**0.961**	**0.940**
	Pub Date	0	101	29	0.000	0.000	0.000	0.000	0.000
	Reference	3,009	235	809	0.742	0.852	0.928	0.788	0.665
	Section	1,154	**143**	373	0.691	0.817	**0.890**	0.756	0.689
	Title	**98**	**0**	**3**	**0.970**	**0.985**	**1.00**	**0.970**	**0.965**
Tabula	Table	101	290	152	0.186	0.314	0.258	0.399	0.190
Unstructured	Caption	647	122	230	0.648	0.786	0.841	0.738	0.606
	Equation	1,340	1,924	1,981	0.255	0.407	0.411	0.403	0.208
	Footer	54	1,133	290	0.037	0.071	0.045	0.157	0.035
	Header	104	412	66	0.179	0.303	0.202	0.612	0.173
	List	**32**	**1789**	**212**	**0.179**	**0.303**	**0.202**	**0.612**	**0.173**
	Reference	2,732	125	1,086	0.693	0.819	0.956	0.716	0.670
	Section	1,343	206	184	0.775	0.873	0.867	0.880	0.727
	Table	**199**	55	**54**	**0.646**	**0.785**	0.783	**0.787**	**0.601**
	Title	91	10	10	0.820	0.901	0.901	0.901	0.811