A Benchmark of PDF Information Extraction Tools using a Multi-Task and Multi-Domain Evaluation Framework for Academic Documents

Norman Meuschke[1,[ORCID]], Apurva Jagdale[2], Timo Spinde[1,[ORCID]],
Jelena Mitrović[2,3,[ORCID]], and Bela Gipp[1,[ORCID]]

[1] University of Göttingen, 37073 Göttingen, Germany
{meuschke, spinde, gipp}@uni-goettingen.de
[2] University of Passau, 94032 Passau, Germany
{apurva.jagdale, jelena.mitrovic}@uni-passau.de
[3] The Institute for Artificial Intelligence R&D of Serbia, 21000 Novi Sad, Serbia

Abstract. Extracting information from academic PDF documents is crucial for numerous indexing, retrieval, and analysis use cases. Choosing the best tool to extract specific content elements is difficult because many, technically diverse tools are available, but recent performance benchmarks are rare. Moreover, such benchmarks typically cover only a few content elements like header metadata or bibliographic references and use smaller datasets from specific academic disciplines. We provide a large and diverse evaluation framework that supports more extraction tasks than most related datasets. Our framework builds upon DocBank, a multi-domain dataset of 1.5M annotated content elements extracted from 500K pages of research papers on arXiv. Using the new framework, we benchmark ten freely available tools in extracting document metadata, bibliographic references, tables, and other content elements from academic PDF documents. GROBID achieves the best metadata and reference extraction results, followed by CERMINE and Science Parse. For table extraction, Adobe Extract outperforms other tools, even though the performance is much lower than for other content elements. All tools struggle to extract lists, footers, and equations. We conclude that more research on improving and combining tools is necessary to achieve satisfactory extraction quality for most content elements. Evaluation datasets and frameworks like the one we present support this line of research. We make our data and code publicly available to contribute toward this goal.

Keywords: PDF · Information Extraction · Benchmark · Evaluation.

1 Introduction

The Portable Document Format (PDF) is the most prevalent encoding for academic documents. Extracting information from academic PDF documents is crucial for numerous indexing, retrieval, and analysis tasks. Document search, recommendation, summarization, classification, knowledge base construction, question answering, and bibliometric analysis are just a few examples [31].

Table 1: Publications on information extraction from PDF documents.

Publication[1]	**Year**	**Task**[2]	**Method**	**Training Dataset**[3]
Palermo [44]	1999	M, ToC	Rules	100 documents
Klink [27]	2000	M	Rules	979 pages
Giuffrida [18]	2000	M	Rules	1,000 documents
Aiello [2]	2002	RO, Title	Rules	1,000 pages
Mao [37]	2004	M	OCR, Rules	309 documents
Peng [45]	2004	M, R	CRF	CORA (500 refs.)
Day [14]	2007	M, R	Template	160,000 citations
Hetzner [23]	2008	R	HMM	CORA (500 refs.)
Councill [12]	2008	R	CRF	CORA (200 refs.), CiteSeer (200 refs.)
Lopez [36]	2009	B, M, R	CRF, DL	None
Cui [13]	2010	M	HMM	400 documents
Ojokoh [42]	2010	M	HMM	CORA (500 refs.), ManCreat FLUX-CiM (300 refs.),
Kern [25]	2012	M	HMM	E-prints, Mendeley, PubMed (19K entries)
Bast [5]	2013	B, M, R	Rules	DBLP (690 docs.), PubMed (500 docs.)
Souza [53]	2014	M	CRF	100 documents
Anzaroot [3]	2014	R	CRF	UMASS (1,800 refs.)
Vilnis [57]	2015	R	CRF	UMASS (1,800 refs.)
Tkaczyk [55]	2015	B, M, R	CRF, Rules, SVM	CiteSeer (4,000 refs.), CORA (500 refs.), GROTOAP, PMC (53K docs.)
Bhardwaj [7]	2017	R	FCN	5,090 references
Rodrigues [49]	2018	R	BiLSTM	40,000 references
Prasad [46]	2018	M, R	CRF, DL	FLUX-CiM (300 refs.), CiteSeer (4,000 refs.)
Jahongir [4]	2018	M	Rules	10,000 documents
Torre [15]	2018	B, M	Rules	300 documents
Rizvi [47]	2020	R	R-CNN	40,000 references
Hashmi [22]	2020	M	Rules	45 documents
Ahmed [1]	2020	M	Rules	150 documents
Nikolaos [33]	2021	B, M, R	Attention, BiLSTM	3,000 documents

[1] Publications in chronological order; the labels indicate the first author only.

[2] (B) Body text, (M) Metadata, (R) References, (RO) Reading order, (ToC) Table of contents

[3] Domain-specific datasets: Computer Science: CiteSeer [43], CORA [39], DBLP [52], FLUX-CiM [10,11], ManCreat [42]; Health Science: PubMed [40], PMC [41]

Only the DocBank dataset by Li et al. [31] offers annotations for 12 diverse content elements in academic documents, including, figures, equations, tables, and captions. Most of these content elements have not been used for benchmark evaluations yet. DocBank is comparably large (500K pages from research papers published on arXiv in a four-year period). A downside of the DocBank dataset is its coarse-grained labels for references, which do not annotate the fields of bibliographic entries like the author, publisher, volume, or date, as do bibliography-specific datasets like unarXive [21] or S2ORC [34].

Table 3 shows PDF information extraction benchmarks performed since 1999. Few such works exist and were rarely repeated or updated, which is sub-optimal given that many tools receive updates frequently. Other tools become technologically obsolete or unmaintained. For instance, pdf-extract[6], lapdftext[7], PDF-SSA4MET[8], and PDFMeat[9] are no longer maintained actively, while ParsCit[10] has been replaced by NeuralParsCit[11] and SciWING[12].

Table 3: Benchmark evaluations of PDF information extraction approaches.

Publication[1]	**Dataset**	**Metrics**[2]	**Tools**	**Labels**[3]
Granitzer [19]	E-prints (2,452 docs.), Mendeley (20,672 docs.)	P, R	2	M
Lipinski [32]	arXiv (1,253 docs.)	Acc	7	M
Bast [6]	arXiv (12,098 docs.)	Custom	14	NL, Pa RO, W
Körner [28]	100 (German docs.)	P, R, F_1	4	Ref
Tkaczyk [54]	9,491 documents	P, R, F_1	10	Ref
Rizvi [48]	8,766 references	F_1	4	Ref

[1] The labels indicate the first author only.
[2] (P) Precision, (R) Recall, (F_1) F_1-score, (Acc) Accuracy
[3] (M) Metadata, (NL) New Line, (Pa) Paragraph, (Ref) Reference, (RO) Reading order, (W) Words

As Table 3 shows, the most extensive dataset used for evaluating PDF information extraction tools so far contains approx. 24,000 documents. This number is small compared to the sizes of datasets available for this task, shown in Table 2. Most studies focused on exclusively evaluating metadata and reference extraction (see also Table 3). An exception is a benchmark by Bast and Korzen

[6] https://github.com/CrossRef/pdfextract
[7] https://github.com/BMKEG/lapdftext
[8] https://github.com/eliask/pdfssa4met
[9] https://github.com/dimatura/pdfmeat
[10] https://github.com/knmnyn/ParsCit
[11] https://github.com/WING-NUS/Neural-ParsCit
[12] https://github.com/abhinavkashyap/sciwing

Table 4: Overview of evaluated information extraction tools.

Tool	**Version**	**Task**[1]	**Technology**	**Output**
Adobe Extract	1.0	G, T	Adobe Sensei AI Framework	JSON, XLSX
Apache Tika	2.0.0	G	Apache PDFBox	TXT
Camelot	0.10.1	T	OpenCV, PDFMiner	CSV, Dataframe
CERMINE	1.13	G, M, R	CRF, iText, Rules, SVM	JATS
GROBID	0.7.0	G, M, R, T	CRF, Deep Learning, Pdfalto	TEI XML
PdfAct	n/a	G, M, R, T	pdftotext, rules	JSON, TXT, XML
PyMuPDF	1.19.1	G	OCR, tesseract	TXT
RefExtract	0.2.5	R	pdftotext, rules	TXT
ScienceParse	1.0	G, M, R,	CRF, pdffigures2, rules	JSON
Tabula	1.2.1	T	PDFBox, rules	CSV, Dataframe

[1] (G) General, (M) Metadata, (R) References, (T) Table

Camelot[18] can extract tables using either the *Stream* or *Lattice* modes. The former uses whitespace between cells and the latter table borders for table cell identification. For our experiments, we exclusively use the Stream mode, since our test documents are academic papers, in which tables typically use whitespace in favor of cell borders to delineate cells. The Stream mode internally utilizes the PDFMiner library[19] to extract characters that are subsequently grouped into words and sentences using whitespace margins.

CERMINE [55] offers metadata, reference, and general extraction capabilities. The tool employs the iText PDF toolkit[20] for character extraction and the Docstrum[21] image segmentation algorithm for page segmentation of document images. CERMINE uses an SVM classifier implemented using the LibSVM[22] library and rule-based algorithms for metadata extraction. For reference extraction, the tool employs k-means clustering, and Conditional Random Fields implemented using the MALLET[23] toolkit for sequence labeling. CERMINE returns a single XML file containing the annotations for an entire PDF. We employ the Beautiful Soup[24] library to filter CERMINE's output files for the annotations relevant to our evaluation.

GROBID[25] [35] supports all four extraction tasks. The tool allows using either feature-engineered CRF (default) or a combination of CRF and DL models realized using the DeLFT[26] Deep Learning library, which is based on TensorFlow and Keras. GROBID uses a cascade of sequence labeling models for different components. The models in the model cascade use individual label sequencing

[18] https://github.com/camelot-dev/camelot

[19] https://github.com/pdfminer/pdfminer.six

[20] https://github.com/itext

[21] https://github.com/chulwoopack/docstrum

[22] https://github.com/cjlin1/libsvm

[23] http://mallet.cs.umass.edu/sequences.php

[24] https://www.crummy.com/software/BeautifulSoup/bs4/doc/

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

[26] https://github.com/kermitt2/delft

The DocBank dataset offers ground-truth annotations for 1.5M content elements on 500K pages. Li et al. extracted the pages from arXiv papers in Physics, Mathematics, Computer Science, and numerous other fields published between 2014 and 2018. DocBank's large size, recency, diversity of included documents, number of annotated content elements, and high annotation quality due to the weakly supervised labeling approach make it an ideal choice for our purposes.

3.3 Evaluation Procedure

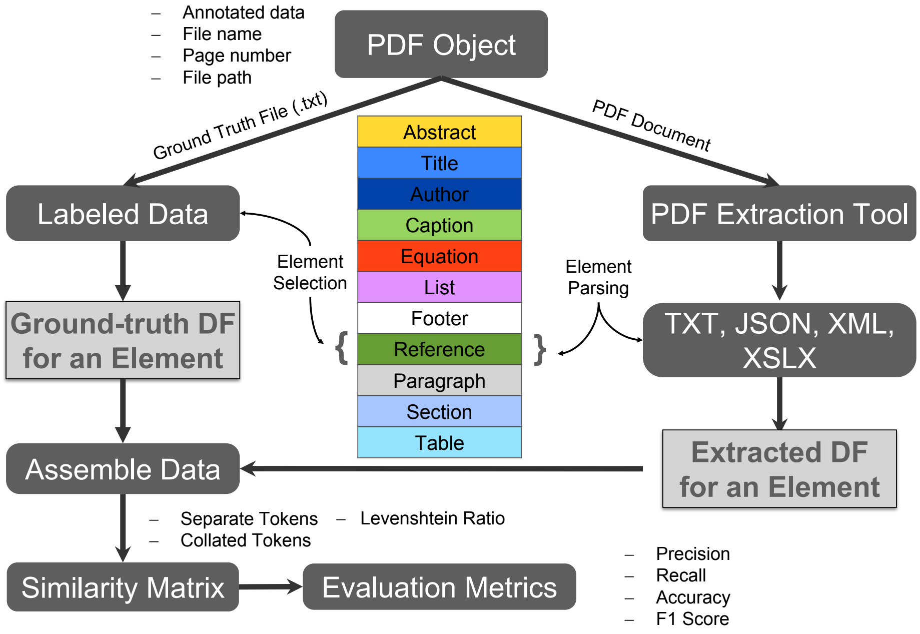


Fig. 3: Overview of the procedure for comparing content elements extracted by IE tools to the ground-truth annotations and computing evaluation metrics.

Figure 3 shows our evaluation procedure. First, we select the PDF files whose associated ground-truth files contain relevant labels. For example, we search for ground-truth files containing *reference* tokens to evaluate reference extraction tools. We include the PDF file, the ground-truth file, the document ID and page number obtainable from the file name (cf. Figure 2), and the file path in a self-defined Python object (see *PDF Object* in Figure 3).

Then, the evaluation process splits into two branches whose goal is to create two pandas data frames—one holding the relevant ground-truth data, and the other the output of an information extraction tool. For this purpose, both the ground-truth files and the output files of IE tools are parsed and filtered for the relevant content elements. For example, to evaluate reference extraction via CERMINE, we exclusively parse reference tags from CERMINE's XML output file into a data frame (see *Extracted DF* in Figure 3).

Finally, we convert both the *ground-truth data frame* and the *extracted data frame* into two formats for comparison and computing performance metrics. The first is the *separate tokens* format, in which every token is represented as a row in the data frame. The second is the *collated tokens* format, in which all tokens are combined into a single space-delimited row in the data frame. Separate tokens serve to compute a strict score for token-level extraction quality, whereas collated tokens yield a more lenient score intended to reflect a tool's average extraction quality for a class of content elements. We will explain the idea of both scores and their computation hereafter.

We employ the *Levenshtein Ratio* to quantify the similarity of extracted tokens and the ground-truth data for both the separate tokens and collated tokens format. Equation (1) defines the computation of the *Levenshtein distance* of the extracted tokens t_e and the ground-truth tokens t_g.

$$lev_{t_e,t_g}(i,j) = \begin{cases} max(i,j), & \text{if } min(i,j) = 0, \\ min \begin{cases} lev_{t_e,t_g}(i-1,j) + 1 \\ lev_{t_e,t_g}(i,j-1) + 1 \\ lev_{t_e,t_g}(i-1,j-1) + 1_{(t_{ei} \neq t_{ej})} \end{cases} & \text{otherwise.} \end{cases} \tag{1}$$

Equation (2) defines the derived Levenshtein Ratio score (γ).

$$\gamma\left(t_e, t_g\right) = 1 - \frac{lev_{t_e,t_g}(i,j)}{|t_e| + |t_g|} \tag{2}$$

Equation (3) shows the derivation of the *similarity matrix* (Δ^d) for a document (d), which contains the Levenshtein Ratio (γ) of every token in the extracted data frame with separate tokens E^s of size m and the ground-truth data frame with separate tokens G^s of size n.

$$\Delta^d_{m\times n} = \gamma\left[E^s_i, G^s_j\right]^{m,n}_{i,j} \tag{3}$$

Using the $m \times n$ similarity matrix, we compute the *Precision* P^d and *Recall* R^d scores according to Equation (4) and Equation (5), respectively. As the numerator, we use the number of extracted tokens whose Levenshtein Ratio is larger or equal to 0.7. We chose this threshold for consistency with the experiments by Granitzer et al. [19]. We then compute the F^d_1 score according to Equation (6) as a token-level score for a tool's extraction quality.

$$P^d = \frac{\#\Delta^d_{i,j} \geq 0.7}{m} \tag{4}$$

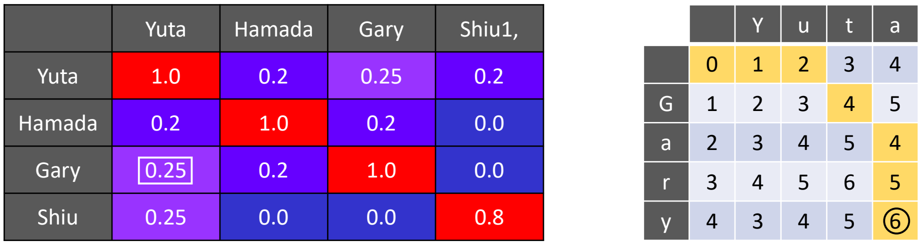
$$R^d = \frac{\#\Delta^d_{i,j} \geq 0.7}{n} \tag{5}$$

$$F_1{}^d = \frac{2 \times P^d \times R^d}{P^d + R^d} \tag{6}$$

Moreover, we compute the *Accuracy* score A^d reflecting a tool's average extraction quality for a class of tokens. To obtain A^d, we compute the Levenshtein Ratio γ of the extracted tokens E^c and ground-truth tokens G^c in the collated tokens format, according to Equation (7).

$$A^d = \gamma\left[E^c, G^c\right] \tag{7}$$

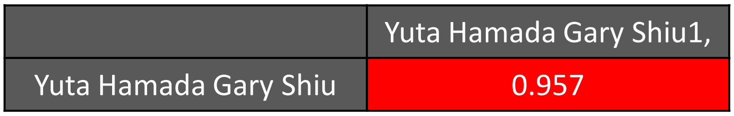
Figure 4 and Figure 5 show the similarity matrices for the author names 'Yuta,' 'Hamada,' 'Gary,' and 'Shiu' using separate and collated tokens, respectively. Figure 4 additionally shows an example computation of the Levenshtein Ratio for the strings *Gary* and *Yuta*. The strings have a Levenshtein distance of six and a cumulative string length of eight, which results in a Levenshtein Ratio of 0.25 that is entered into the similarity matrix. Figure 5 analogously exemplifies computing the Accuracy score of the two strings using collated tokens.



	Yuta	Hamada	Gary	Shiu1,
Yuta	1.0	0.2	0.25	0.2
Hamada	0.2	1.0	0.2	0.0
Gary	0.25	0.2	1.0	0.0
Shiu	0.25	0.0	0.0	0.8

		Y	u	t	a
	0	1	2	3	4
G	1	2	3	4	5
a	2	3	4	5	4
r	3	4	5	6	5
y	4	3	4	5	6

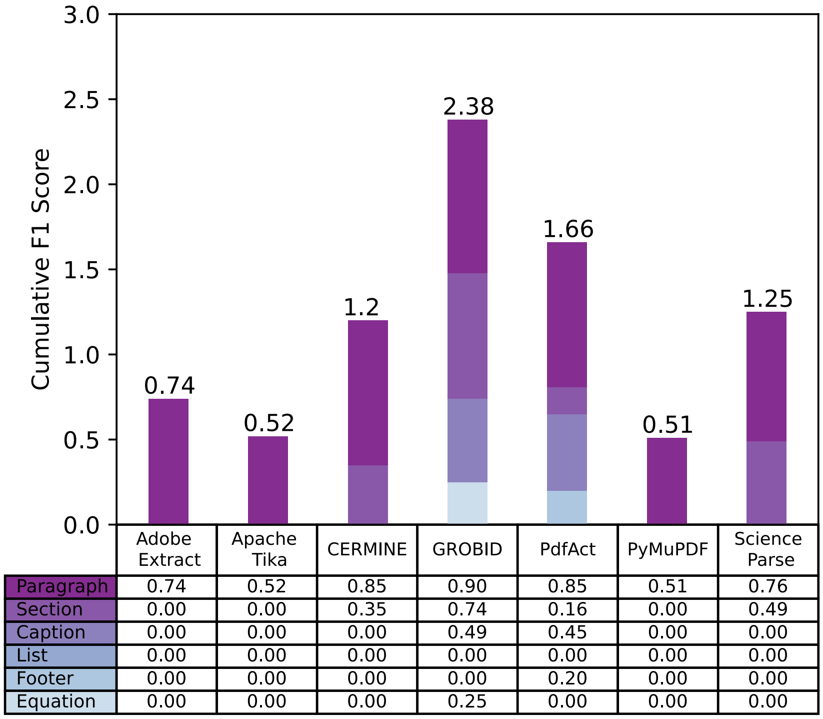
Fig. 4: Left: Similarity matrix for author names using separate tokens.
Right: Computation of the Levenshtein distance (6) and the optimal edit transcript (yellow highlights) for two author names using dynamic programming.



	Yuta Hamada Gary Shiu1,
Yuta Hamada Gary Shiu	0.957

Fig. 5: Similarity matrix for two sets of author names using collated tokens.

For the general extraction task, GROBID outperforms other tools due to its segmentation model[43], which detects the main areas of documents based on layout features. Therefore, frequent content elements like paragraphs will not impact the extraction of rare elements from a non-body area by keeping the imbalanced classes in separate models. The cascading models used in GROBID also offer the flexibility to tune each model. Using layouts and structures as a basis for the process allows the association of simpler training data.



	Adobe Extract	Apache Tika	CERMINE	GROBID	PdfAct	PyMuPDF	Science Parse
Paragraph	0.74	0.52	0.85	0.90	0.85	0.51	0.76
Section	0.00	0.00	0.35	0.74	0.16	0.00	0.49
Caption	0.00	0.00	0.00	0.49	0.45	0.00	0.00
List	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Footer	0.00	0.00	0.00	0.00	0.20	0.00	0.00
Equation	0.00	0.00	0.00	0.25	0.00	0.00	0.00

Fig. 9: Results for general data extraction.

The breakdown of results by tools shown in Table 6 underscores the main takeaway point of the results' presentation for the individual extraction tasks. The tools' results differ greatly for different content elements. Certainly, no tool performs best for all elements, rather, even tools that perform well overall can fail completely for certain extraction tasks. The large amount of content elements whose extraction is either unsupported or only possible in poor quality indicates a large potential for improvement in future work.

[43] https://grobid.readthedocs.io/en/latest/Principles/

Table 6: Results grouped by extraction tool.

Tool[1]	**Label**	**# Detected**	**# Processed**[2]	*Acc*	F_1	*P*	*R*
Adobe Extract	Table	1,635	736	**0.52**	**0.47**	**0.45**	**0.49**
	Paragraph	3,985	3,088	0.85	0.74	0.72	0.76
Apache Tika	Paragraph	339,603	258,582	0.55	0.52	0.43	0.65
Camelot	Table	16,289	**11,628**	0.27	0.30	0.23	0.44
CERMINE	Title	16,196	14,501	0.84	0.81	0.81	0.81
	Author	**19,788**	14,797	0.43	0.44	0.44	0.46
	Abstract	19,342	16,716	0.71	0.72	0.68	0.76
	Reference	**40,333**	35,193	0.80	0.74	0.71	0.77
	Paragraph	361,273	348,160	0.89	0.85	0.83	0.87
	Section	**163,077**	139,921	0.40	0.35	0.32	0.38
GROBID	Title	16,196	16,018	**0.92**	**0.91**	**0.91**	**0.92**
	Author	**19,788**	**19,563**	**0.54**	**0.52**	**0.52**	**0.53**
	Abstract	19,342	**18,714**	0.82	**0.82**	**0.81**	0.83
	Reference	**40,333**	**36,020**	**0.82**	**0.79**	**0.79**	**0.80**
	Paragraph	361,273	**358,730**	**0.90**	**0.90**	**0.89**	**0.91**
	Section	**163,077**	**163,037**	**0.77**	**0.74**	**0.73**	**0.76**
	Caption	**90,606**	**62,445**	**0.57**	**0.49**	**0.47**	0.51
	Table	**16,740**	8,633	0.24	0.23	0.23	0.23
	Equation	**142,736**	**96,560**	**0.26**	**0.25**	**0.20**	**0.32**
PdfAct	Title	**17,670**	**16,834**	0.85	0.85	0.85	0.86
	Author	13,110	2,187	0.14	0.13	0.12	0.18
	Abstract	**21,470**	4,683	0.17	0.16	0.15	0.20
	Reference	30,263	12,705	0.19	0.15	0.17	0.20
	Paragraph	**361,318**	357,905	0.85	0.85	0.80	0.89
	Section	129,361	87,605	0.21	0.16	0.12	0.25
	Caption	83,435	53,314	0.45	0.45	0.40	**0.52**
	Footer	**32,457**	**26,252**	**0.23**	**0.20**	**0.25**	**0.16**
PyMuPDF	Paragraph	339,650	258,383	0.55	0.51	0.41	0.65
RefExtract	Reference	**40,333**	38,405	0.55	0.49	0.44	0.55
Science Parse	Title	11,696	11,687	0.79	0.70	0.70	0.70
	Author	471	471	**0.54**	**0.52**	**0.52**	**0.53**
	Abstract	14,150	14,149	**0.83**	0.81	0.73	**0.90**
	Reference	**40,333**	35,200	0.55	0.49	0.49	0.50
	Paragraph	**361,318**	355,529	0.79	0.76	0.76	0.76
	Section	**163,077**	158,556	0.54	0.49	0.49	0.50
Tabula	Table	10,361	9,456	0.29	0.28	0.20	0.46

[1] Boldface indicates the best value for each content element type.

[2] The differences in the number of detected and processed items are due to PDF Read Exceptions or Warnings. We label an item as processed if it has a non-zero F_1 score.

5 Conclusion and Future Work

We present an open evaluation framework for information extraction from academic PDF documents. Our framework uses the DocBank dataset [31] offering 12 types and 1.5M annotated instances of content elements contained in 500K pages of arXiv papers from multiple disciplines. The dataset is larger, more topically diverse, and supports more extraction tasks than most related datasets.

We use the newly developed framework to benchmark the performance of ten freely available tools in extracting document metadata, bibliographic references, tables, and other content elements in academic PDF documents. GROBID, followed by CERMINE and Science Parse achieves the best results for the metadata and reference extraction tasks. For table extraction, Adobe Extract outperforms other tools, even though the performance is much lower than for other content elements. All tools struggle to extract lists, footers, and equations.

While DocBank covers more disciplines than other datasets, we see further diversification of the collection in terms of disciplines, document types, and content elements as a valuable task for future research. Table 2 shows that more datasets suitable for information extraction from PDF documents are available but unused thus far. The weakly supervised annotation approach used for creating the DocBank dataset is transferable to other LaTeX document collections.

Apart from the dataset, our framework can incorporate additional tools and allows easy replacement of tools in case of updates. We intend to update and extend our performance benchmark in the future.

The extraction of tables, equations, footers, lists, and similar content elements poses the toughest challenge for tools in our benchmark. In recent work, Grennan et al.[20] showed that the usage of synthetic datasets for model training can improve citation parsing. A similar approach could also be a promising direction for improving the access to currently hard-to-extract content elements.

Combining extraction approaches could lead to a one-fits-all extraction tool, which we consider desirable. The Sciencebeam-pipelines[44] project currently undertakes initial steps toward that goal. We hope that our evaluation framework will help to support this line of research by facilitating performance benchmarks of IE tools as part of a continuous development and integration process.

References

1. Ahmed, M.W., Afzal, M.T.: FLAG-PDFe: Features Oriented Metadata Extraction Framework for Scientific Publications. IEEE Access **8**, 99458–99469 (May 2020). https://doi.org/10.1109/ACCESS.2020.2997907
2. Aiello, M., Monz, C., Todoran, L., Worring, M.: Document Understanding for a Broad Class of Documents. International Journal on Document Analysis and Recognition **5(1)** (Aug 2002). https://doi.org/10.1007/s10032-002-0080-x
3. Anzaroot, S., Passos, A., Belanger, D., McCallum, A.: Learning Soft Linear Constraints with Application to Citation Field Extraction. In: Proceedings of the 52nd

[44] https://github.com/elifesciences/sciencebeam-pipelines

55. Tkaczyk, D., Szostek, P., Fedoryszak, M., Dendek, P.J., Bolikowski, Ł.: CERMINE: automatic extraction of structured metadata from scientific literature. International Journal on Document Analysis and Recognition (IJDAR) **18**(4), 317–335 (Dec 2015). https://doi.org/10.1007/s10032-015-0249-8
56. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is All You Need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. pp. 6000–6010. NIPS'17, Curran Associates Inc., Red Hook, NY, USA (2017), https://dl.acm.org/doi/10.5555/3295222.3295349
57. Vilnis, L., Belanger, D., Sheldon, D., McCallum, A.: Bethe Projections for Non-Local Inference. In: Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence. pp. 892–901. UAI'15, AUAI Press, Arlington, Virginia, USA (2015). https://doi.org/10.48550/arXiv.1503.01397