
ADAPARSE: AN ADAPTIVE PARALLEL PDF PARSING AND RESOURCE SCALING ENGINE

**Carlo Siebeneschuh^{1,2} Kyle Hippe^{1,2} Ozan Gokdemir^{1,2} Alexander Brace^{1,2} Arham Khan¹ Khalid Hossain²
Yadu Babuji² Nicholas Chia² Venkatram Vishwanath² Rick Stevens^{1,2} Arvind Ramanathan^{1,2} Ian Foster^{1,2}
Robert Underwood²**

ABSTRACT

Language models for scientific tasks are trained on text from scientific publications—most distributed as PDFs that require parsing. PDF parsing approaches range from inexpensive heuristics (for simple documents) to computationally intensive ML-driven systems (for complex or degraded ones). The choice of the “best” parser for a particular document depends on 1) its computational cost and 2) the accuracy of its output. To address these issues, we introduce an Adaptive Parallel PDF Parsing and Resource Scaling Engine (AdaParse), a data-driven strategy for assigning an appropriate parser to each document. We enlist scientists to select preferred parser outputs and incorporate this information through direct preference optimization (DPO) into AdaParse, thereby aligning its selection process with human judgment. AdaParse then incorporates hardware requirements and (aligned) predicted accuracy of each parser to orchestrate computational resources efficiently for large-scale parsing campaigns. We demonstrate that AdaParse, when compared to state-of-the-art parsers, improves throughput by $17\times$ while still achieving comparable accuracy (actually, 0.2% better) on a benchmark set of 1000 scientific documents. AdaParse’s combination of high accuracy and parallel scalability makes it feasible to parse large-scale scientific document corpora to support the development of high-quality, trillion-token-scale text datasets.

The implementation is available at <https://github.com/7shoe/AdaParse/>.

1 INTRODUCTION

The great wealth of information stored in the scientific literature and the successes of large language models (LLMs) motivate efforts to train science-specialized LLMs on scientific documents (Beltagy et al., 2019; Taylor et al., 2022). However such efforts require immense amounts of text for training (Chowdhery et al., 2022; Li et al., 2024), and much of it is represented in Portable Document Format (PDF). Exploiting this text requires correctly parsing information from PDFs, which is challenging due to their print-focused layout-based structure that is not designed for machine readability (Coulon et al., 2023). For complex PDFs, lightweight parsers often extract text swiftly but incorrectly, introducing artifacts that degrade the performance of LLMs trained on it. Mitigating adverse effects requires substantially more training data to achieve the same final LLM performance, further exacerbating the challenge (Sorscher et al., 2023).

¹Department of Computer Science, University of Chicago, Chicago, Illinois, USA ²Argonne National Laboratory, Lemont, Illinois, USA. Correspondence to: Carlo Siebeneschuh <siebeneschuh@uchicago.edu>.

Proceedings of the 8th MLSys Conference, Santa Clara, CA, USA, 2025. Copyright 2025 by the author(s).

As we will show, state-of-the-art high-quality parsing software is extremely computationally demanding, being able to parse only 1–2 PDF/s on a node with 4 A100 GPUs, making them impractical for datasets of hundreds of millions of scientific papers (Figure 3). Unsurprisingly, existing datasets of scientific tokens suitable for LLM training are modest in size (e.g., the popular Dolma dataset contains only 70B tokens from scientific sources (Soldaini et al., 2024)) and often suffer from poor parse quality (Bast & Korzen, 2017). Thus *accurate* and *efficient* PDF parsing is a central problem for those seeking to build high-quality AI-based scientific assistants and similar tools.

However, not all is lost. As we will show, many “simpler” documents can be parsed with lightweight tools orders of magnitude faster with similar (or even improved) output quality as compared to their compute-intensive counterparts. We leverage this fact to develop an adaptive parsing strategy that invokes lightweight parsers on simpler documents while reserving high-quality parsers for those deemed complex—thus deploying the most promising parser for each particular PDF in a way that balances competing demands for accuracy and throughput. We show that this approach can greatly improve overall goodput as measured by *accepted* textual tokens generated per resource unit.

To realize these benefits, this work makes the following contributions:

- A comprehensive benchmark to assess parser performance characteristics, conducted on 25,000 PDFs from across disciplines and publishers, including an assessment of how human perception and parser output quality align with commonly used metrics.
- A predictive algorithm that, for any PDF, selects the parser most likely to yield accurate output, adapting to document attributes and user preferences.
- The integrated design of batching, prefetching, parallel execution, and scheduling to realize adaptive parsing of PDFs for high throughput and quality on leadership-class HPC systems.

The remainder of the paper is organized as follows: In Section 2, we describe parsing challenges and failure modes, how quality is compared between parsers, and why PDF parsing presents a non-trivial parallel and distributed systems problem. Next, in Section 3, we describe the various classes of parsers and how they can be used in parallel workflows to parse PDFs *en masse*. After that, we formulate our task of producing high-quality text output as an optimization problem in Section 4. We then provide an overview of our system in Section 5 and present the key optimizations that we employ to achieve both high throughput and high-quality text. We discuss our experimental methodology in Section 6. Finally, we present our evaluation in Section 7, followed by conclusions and future work in Section 8.

2 BACKGROUND

2.1 Challenges in PDF Parsing

Document parsers can fail to produce accurate text output in a variety of ways. As illustrated in Figure 1, failures can include introducing whitespace, substituting words, scrambling characters or words, corrupting identifiers or references, or even dropping entire pages. Notably, such errors are not confined to lightweight parsers; even sophisticated parsing software can encounter them. In fact, we have found that the most severe failure mode—dropping an entire page—occurs with the parser that otherwise delivers the most accurate results.

These failures are driven by the fact that the PDF is layout-driven: it is designed to provide versatility in achieving a desired visual appearance. This versatility means that parsing even a single PDF document can be challenging. Born-digital PDFs can contain diverse elements such as figures, tables, and rich media like videos (Corrêa & Zander, 2017). They may even hold hidden information or malware (Kuribayashi & Wong, 2021; Singh et al., 2020). Scanned

documents, on the other hand, suffer from significant quality degradation, complicating content extraction (Mujumdar et al., 2019). This diversity virtually rules out the development of a universal parsing strategy, as a one-size-fits-all parser would either be too simplistic to handle complex cases or inefficient in parsing simpler ones. This situation necessitates the use of an *adaptive* parsing strategy able to address each PDF individually. Yet naively applying each available parser to a given document and selecting the output that appears the most accurate is infeasible. Thus, parser selection must be based on easily available data and form a prediction on it (Ravi et al., 2008).

2.2 Evaluation of Parsing Accuracy

A key obstacle to evaluating PDF parsers is the lack of a quality metric for measuring their output against groundtruth text. In the absence of a universal accuracy measure that comprehensively captures the similarity between long-form texts—accounting for syntax, spelling errors, and scientific content—several proxy metrics are used.

Traditional metrics assess the similarity between parser output and groundtruth text on a character level. For example, the Levenshtein distance reports the minimum number of character edits required to transform one text into the other (Levenshtein, 1966). Although straightforward to compute, it may poorly align with human perception of quality (Nerbonne et al., 1999). Moreover, these routines can prove computationally prohibitive for ultra-long text sequences as encountered in parsed PDF text. Moreover, scientific (in-)accuracy goes beyond character errors that may prove subtle but deadly. For example, while the edit distance between “hyperthyroidism” and “hypothyroidism” is just two, implying a normalized similarity of 86.7%, the treatments for these conditions are opposites. Similarly, changes in character capitalization can turn the measure of acidity (pH) into the phenyl group (Ph).

Modern metrics such as BLEU (Bilingual Evaluation Understudy) (Papineni et al., 2002) and ROUGE (Recall-Oriented Understudy for Gisting Evaluation) (Lin, 2004) set out to measure string similarity in a manner more aligned with human perception. These metrics are based on the number of matching n-grams and capture meaning across multiple words. Regardless, they may still fail to capture scientific meaning even when evaluating a single sentence. For instance, consider the following groundtruth text:

“The gravitational force between two masses is directly proportional to the product of their masses and inversely proportional to the square of the distance between them.”

and the candidate text:

“The gravitational force inversely masses the proportional distance between two products and is directly proportional to the square of objects.”

When evaluating under BLEU and ROUGE, we observe 0.32 and 0.82 respectively—indicating reasonable and high accuracy to the groundtruth text, despite the incoherent and factually erroneous candidate text.

Furthermore, these metrics are designed to assess the sentence-length quality of neural translations rather than the multi-page parser output of scientific documents (Graham, 2015). The accumulation of seemingly small mistakes can result in text that appears to be of high quality but significantly distorts the intended insights. Finally, current parsers require hyperparameters that are only tacitly assumed fixed and can hardly be considered canonical (Post, 2018). While indicative of perceived text quality, these metrics are insufficient to thoroughly compare PDF parser output against the groundtruth text.

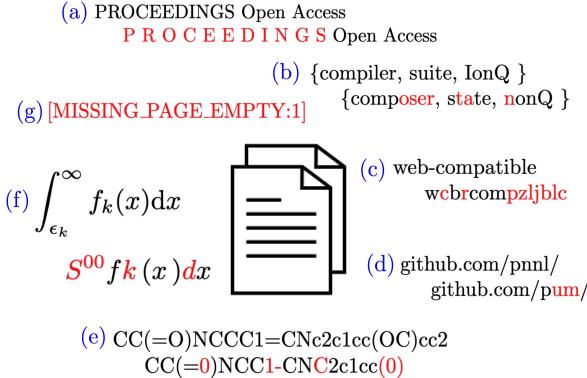


Figure 1. Failure modes of PDF parsers, (a) whitespace injection, (b) word substitution, (c) character scrambling, (d) character substitution, (e) corrupted SMILES, (f) LaTeX to plaintext conversion, (g) document page dropped.

2.3 Datasets and Benchmarks

Several datasets have been created to assess or improve PDF parsing technology (Jimeno Yepes et al., 2021). Early datasets focused on specific applications such as license plate or business card recognition (Bulan et al., 2017; Saiga et al., 1993). Recent datasets, on the other hand, cater to specific document types, including handwritten notes, scanned documents, and layout-rich publications (Shaffi & Hajamohideen, 2021; Jaume et al., 2019; Zhu et al., 2022; Paudel et al., 2024). A recent survey indicates that scientific documents are exceptionally challenging for parsers (Adhikari & Agarwal, 2024).

S2ORC (Lo et al., 2019) is particularly relevant to scientific information parsing as it contains over 8 million full-text academic papers from diverse publishers. It is not suitable

for this work, however, as groundtruth text was synthesized by a PDF parser (GROBID), and the resulting text/PDF pairs served as the training data of another (Nougat) (Blecher et al., 2023). Maintaining the integrity of our benchmark requires the use of text and PDF pairs that have not been incorporated into the training of neural networks deployed by PDF parsers. However, recent research continues to prioritize dataset size over the quality of annotations or access to closed-source content. As AI-driven parsing systems evolve, it becomes increasingly important that their evaluation undergoes equally rigorous, AI-level scrutiny to ensure reliability and accuracy (Paudel et al., 2024).

2.4 Parallel Systems

For a given parser, runtime depends on the content of a PDF, which can include vector graphics, raster images, or even multimedia elements. Runtimes vary even more widely across parsing strategies, with some parsers employing large machine learning (ML) models to process documents line by line. Since it is a priori unknown what parser is best to handle a given PDF, the overall time to parse text is subject to a great deal of uncertainty.

Writing software to parse a single PDF is tedious, but scaling that software to 100 million PDFs is a considerable challenge for parallel and distributed systems. Parsing involves I/O-intensive workloads, where large batches of PDFs are read into memory and substantial text data is written to distributed storage. Heterogeneous PDFs lead to varying batch sizes and uneven processing times, complicating load balancing across distributed nodes. A resilient infrastructure is necessary to handle corrupted PDFs that may be present in datasets, and potential security risks can arise if PDFs contain malicious software. Additionally, indexing the parsed text is challenging, as the lack of consistency across documents complicates maintaining accurate metadata.

3 RELATED WORK

3.1 Parsers

We can distinguish two classes of PDF parsing methods: text extraction and text recognition. Text recognition, in turn, includes both traditional optical character recognition (OCR) techniques and newer approaches that leverage modern ML models, such as Vision Transformers (ViT).

3.1.1 Extraction

Text extraction tools retrieve content directly from the textual layer embedded within a PDF. MuPDF, for example, is a high-performance extraction and rendering tool (Artifex Software). Its Python binding PyMuPDF supports various input and output file types, such as LlamaIndex, a format tailored to LLM data curation. Another popular extraction

tool, pypdf (Fenniak et al., 2024), is a pure Python library.

Extraction tools are generally fast and language-agnostic, indiscriminately retrieving the entirety of text embedded within a document. However, they falter when text is either not embedded explicitly or is of poor quality. Text scrambling is sometimes employed by authors to obstruct extraction. Even if the text is embedded with good intentions, it may still be of low quality if initially inferred and attached by subpar text recognition software.

3.1.2 Optical Character Recognition (OCR)

Text recognition addresses these challenges by converting images of text into machine-readable formats. Optical character recognition employs computer vision techniques to transcribe characters line-by-line. OCR is commonly applied to scanned documents to create an explicit text layer. Numerous libraries support transformation of documents into searchable, structured text (Neudecker et al., 2021).

Tesseract is an open-source OCR engine that has been refined over four decades, predating the PDF itself (Smith, 2007). Now in its fifth version, it employs long short-term memory networks (LSTMs) to infer text sequences from image input. However, its ability to adapt these models to specific document corpora has been limited as training functionality is no longer supported. GROBID (GeneRation Of BIbliographic Data) is another tool that combines machine learning with text extraction to generate highly structured outputs (GRO, 2008–2025). GROBID natively provides some parallel parsing capabilities via multi-threading. It is particularly well-suited for scientific document parsing, offering features such as references, affiliations, and metadata extraction. GROBID is flexible, utilizing entity-specific ML models for bibliographic data extraction and large language models (LLMs) for text completion. It represents a growing trend toward blending classical OCR with modern ML tools (Lopez, 2009).

OCR is generally robust in parsing text from documents as it does not rely on an embedded text layer. However, OCR is computationally intensive, often operating orders of magnitude slower than text extraction tools. Throughput can be further reduced by the need for post-processing to properly format the extracted text (Nguyen et al., 2021). Finally, OCR models often require training or calibration for optimal performance. Unsurprisingly, modern OCR implementations frequently rely on GPUs for improved efficiency (Du et al., 2020).

3.1.3 Vision Transformers (ViTs)

Recent innovations have led to the development of Transformer-based neural architectures for OCR (Li et al., 2023). Such Vision Transformers (ViTs) learn to decode

text from page images in an end-to-end manner. The Document Understanding Transformer (Donut) pioneered this approach for document text recognition, initially focusing on receipts (Kim et al., 2022). Nougat and μ gat extended these capabilities to the parsing of scientific PDFs (Quattrini et al., 2024). Marker further refines this approach through explicit layout detection that precedes parsing of individual document elements through texify (Lab, 2024).

Vision Transformers have shown significant promise in parsing scientific documents. They excel at navigating layout-dense PDF pages and are specifically trained to decode LaTeX equations. However, ViTs are highly compute-intensive at inference time, with their runtime scaling quadratically in the number of image patches. Even with vast datasets of PDF and groundtruth text pairs, along with the computational power required for training, their ability to generalize to unseen document types remains uncertain—particularly in the absence of properly held-out benchmark data.

3.2 Adaptive Parsing

A substantial body of research focuses on training and applying neural networks to tasks involving the prediction of various accuracy metrics. The use of data-driven models to predict a document parser’s performance, or to select an appropriate parser through classification, is not an entirely novel concept. For instance, methods have been developed to estimate parser accuracy based on document metadata (Ravi et al., 2008). Other approaches have explored selective content parsing (Zuidema, 2007) or training parser ensembles via bootstrapping (Steedman et al., 2003). However, much of this earlier work centered on short text inputs (e.g., sentences), predating the rise of large language models and advances in the classification and regression of long-form text, which can now be leveraged to more effectively predict optimal parser candidates.

3.3 Scientific Corpora

Despite the wide range of datasets for training LLMs, there are just two major sources of scientific articles—PILE (Gao et al., 2020) (which contains, for example, ArXiV in L^AT_EX form) and S2ORC (Lo et al., 2019)—that are used in the training of open LLMs. However, these two sources can be vastly under-inclusive of scientific documents. From obtaining access to collections such as the ACM Digital Library and comparing them to these sources, we have measured that as many as 80% of scientific documents from publishers like ACM are not contained in PILE and S2ORC, presenting a gap that would be addressed through adaptive and high-quality PDF parsing.

The PILE dataset includes a subset called PhilPapers, which consists of academic PDFs parsed using Apache PDFBox (Gao et al., 2020). Regardless, the amount of scientific

content is relatively small and almost exclusively sourced from L^AT_EX sources of ArXiV rather than PDFs. While parsing from L^AT_EX can be more reliable than parsing from PDFs, L^AT_EX sources are not accessible for most papers.

Semantic Scholar’s S2ORC constitutes the other extensive, open-source dataset of scientific documents (Lo et al., 2019). Many other collections that incorporate scientific papers rely on S2ORC for their scientific collections, including the Dolma (Soldaini et al., 2024) and the RedPajama family of datasets (Elazar et al., 2023).

Since data curation is crucial for training ever-larger language models, it is likely that leading companies such as OpenAI, Meta, Google, and Mistral have developed proprietary parsing tools to handle this task. Microsoft’s Donut and Meta’s Nougat demonstrate their capability to do so. Nevertheless, specific tools, document collections, and computational scales remain undisclosed.

4 PROBLEM STATEMENT

An ideal parsing strategy will maximize the accuracy of text output while minimizing the computational cost of obtaining it. We formalize this notion to design AdaParse that optimally balances accuracy and runtime considerations.

4.1 Accuracy

Consider d_i to be a PDF document that is identified by an index $i \in [n]$ and spans pages of text, tables, and figures. Denote the associated (groundtruth) text by $\psi_i = \psi(d_i) \in \Sigma^*$, a sequence of characters over an alphabet Σ (e.g., Unicode). While the alphabet is usually known, the document’s groundtruth text is not directly observable and must be approximated by a parser.

There is a set of parsers $\{\phi_1, \dots, \phi_m\}$ available to retrieve text from a document collection $\{d_1, \dots, d_n\}$. Invoking a parser ϕ_j on a PDF document d_i provides an approximation of its groundtruth text $\phi_j(d_i) \approx \psi_i$. The quality of the approximation can be assessed by an accuracy metric \mathcal{A} that may be defined by

$$\mathcal{A}(\phi_j, d_i) = a(\|\phi_j(d_i) - \psi_i\|_{\Sigma^*})$$

through some norm over the alphabet $\|\cdot\|_{\Sigma^*}$ and a monotonically decreasing function a mapping the distance of the strings to a quality score. The accuracy measure is abstract not because of mathematical convenience but due to the nature of document parsing: It is unclear what type of dissimilarity best captures scientifically sound and faithful parser text output.

The computational resources (e.g., runtime) required to parse a document are given by $\mathcal{T}_k(\phi, d)$. The resource usage for $k \in \{CPU_{mem}, CPU_{time}, GPU_{mem}, GPU_{time}\}$ de-

pends on the document, parser, and system used.

We formalize the trade-off between accuracy and efficiency by assigning any of the m parsers to each of the n documents individually, i.e., $j_i \in [m]$, and optimize the following conflicting objectives. For any assignment of parsers to the dataset of documents $\mathbf{j} = (j_1, \dots, j_n) \in [m]^n$ we want to maximize overall accuracy

$$\max_{\mathbf{j}} \left\{ \sum_{i=1}^n \mathcal{A}(\phi_{j_i}, d_i) \right\}$$

while simultaneously minimizing total computational cost

$$\min_{\mathbf{j}} \left\{ \sum_{i=1}^n \mathcal{T}_k(\phi_{j_i}, d_i) \right\}.$$

Imposing a constraint on one objective while optimizing the other strikes a balance. Since accuracy is not observable, we aim to maximize its conditional expectation by selecting the appropriate parser ϕ_{j_i} . The expectation is conditioned on the document d_i ’s first page’s text $\phi_1^1(d_i)$ parsed by the default parser ϕ_1 . The resulting constrained optimization problem

$$\begin{aligned} \max_{\mathbf{j}} & \left\{ \sum_{i=1}^n \mathbb{E} [\mathcal{A}(\phi_{j_i}, \psi_i) \mid \phi_1^1(d_i)] \right\} \\ \text{s.t. } & \sum_{i=1}^n \mathcal{T}(\phi_{j_i}, d_i) \leq \bar{\mathcal{T}} \end{aligned}$$

conveys a crucial property. We can partition the dataset into subsets of $\{n_1, \dots, n_L\}$ documents such that $\sum_{i=1}^L n_i = n$ and process them across L nodes. If each node $l \in [L]$ adheres to a computational budget of $\frac{n_l}{n} \bar{\mathcal{T}}$, the overall required resources will not exceed $\bar{\mathcal{T}}$. Therefore, adaptive document parsing with heterogeneous parsing algorithms can be realized through embarrassingly parallel workloads. A tuning parameter $\alpha \in [0, 1]$ controls the trade-off between (expected) accuracy and runtime in AdaParse.

4.2 Direct Preference Optimization

While parsing accuracy and its trade-off with efficiency appear vague, scientists usually have a strong preference when they are faced with different parser outputs of the same document $\phi_1(d_i)$ and $\phi_2(d_i)$, irrespective of whether groundtruth text ψ_i is available. Therefore, instead of fixing an accuracy measure to $\mathcal{A} =$ ROUGE, for example, we attempt to (implicitly) learn one through user preferences. This is not far-fetched, as accuracy measures like BLEU or ROUGE are designed to be strongly correlated with human preferences (Reiter, 2018). Since a (predicted) accuracy measure only serves as a means to assign a parser to a document, we allow

the predictive model to learn this assignment directly from user input through direct preference optimization (DPO).

Connecting user preference with predicted accuracy has a practical rationale. Malformed text in the parser output $\phi_j(d_i)$ is indicative of overall parser quality. Moreover, there are specific patterns that strongly inform accuracy estimates and human perception of quality alike; see Figure 1. Training a model to infer accuracy from the presence of such malformed text patterns offers a foothold to learn to select a parser adaptively.

In principle, a model capable of assigning a scalar to text, i.e., $\pi_\theta : \Sigma^* \rightarrow [0, 1]$, is a potential candidate for inferring (normalized) accuracy. Rule-based approaches or classical ML models offer interpretable and tractable solutions. Expressive models such as LLMs, on the other hand, that were pre-trained on broad textual data, can be fine-tuned in text sequence regression to make a prediction on text accuracy based on subtle features.

Given a sufficient dataset, a model π_θ can be fine-tuned to predict the BLEU accuracy. This is the crucial ingredient to allow a parsing strategy to predict a good parser-document matching. Moreover, recent strategies for aligning LLMs with human preferences can allow such a model to infer an accuracy measure (implicit in the model).

5 DESIGN

5.1 Overview: Why Adaptive Parsing

The empirical results indicate that the versatility of the layout and textual content of PDF documents prevents assigning a likely parser through deterministic rules alone. For example, the scientific category to which a document belongs (as indicated by associated keywords) is only a weak indicator of its actual content and the difficulty of parsing it. For instance, a research paper on machine learning may boast hundreds of LaTeX expressions, more akin to a mathematics paper. Similarly, document metadata such as the publication year can fail to represent the quality of the embedded text, as that text may have been attached with state-of-the-art OCR software long after the document’s publication. Regardless, obtaining any of these features requires parsing the document. In turn, choosing the optimal parser for a document appears to require parsing it beforehand.

We cut this Gordian knot by leveraging text extraction to inform if and what subsequent text recognition algorithm (OCR or ViT) should be run. In particular, PyMuPDF offers exceedingly fast text extraction, with a throughput $135\times$ higher than Nougat and $13\times$ greater than that of pypdf. Thus prefacing parser selection with it is computationally cheap. Furthermore, the lower accuracy of PyMuPDF works partially in our favor: Malformed substrings (e.g., of La-

TeX equations, whitespace, or scrambled characters) that are typical of text extraction output are informative for the predictive parser selection algorithm.

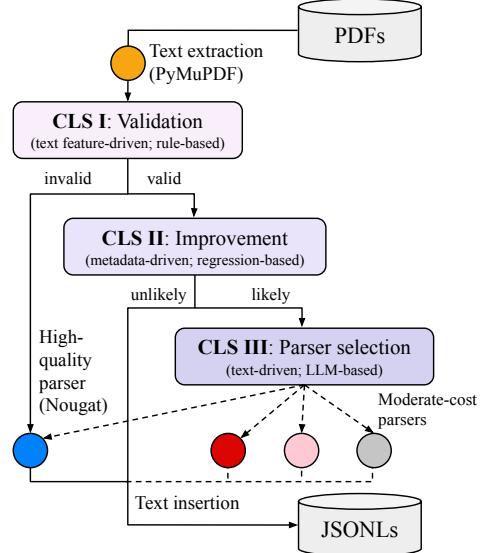


Figure 2. System architecture diagram for a range of predictive models: After an initial text extraction step (PyMuPDF), PDFs are routed through a hierarchical classification pipeline. **CLS I** predicts the binary quality attribute of the extracted text through coarse but fast-to-compute features (e.g., text length). For valid texts, **CLS II** assesses if an improvement is likely for any other parser. If affirmative, **CLS III** selects the parser most likely to improve output text quality.

Parser selection can be performed as a hierarchical classification scheme based on the PyMuPDF-extracted text. The first classification stage CSL I employs aggregate statistics computed from the extracted text (e.g., number of characters) to infer validity. While simplistic, the features are highly interpretable and permit rapid inference. If the PyMuPDF text is deemed invalid, the PDF is sent to the high-quality Nougat parser. If, however, the text is deemed valid, a second classification stage CSL II is applied to determine if parsing with another parser (including Nougat) may nevertheless bring a significant improvement in parse quality. This binary label is inferred from metadata (e.g., authoring tool, year of publication, number of pages). If a significant improvement is deemed unlikely, the PyMuPDF-extracted text is accepted as the document parse and is subsequently written to storage. On the other hand, if improvement is predicted as likely, the third classification stage CSL III is applied to select the parser. Since this decision is based on subtle patterns in the extracted text, a fine-tuned LLM is invoked for this multi-class downstream task.

We introduce two implementations. The first variant, **AdaParse (FT)**, implements the classification stages **CLS I** and **CLS II** within a single routine. If an improvement appears

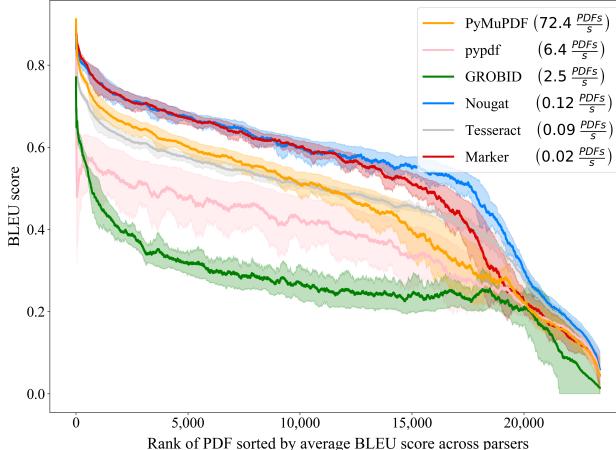


Figure 3. Parser performance (BLEU) for $n = 23,398$ PDFs. They are sorted by parsing difficulty which is estimated for each document by the average BLEU score across parsers. The higher the rank, the greater the estimated parsing difficulty. Throughputs for a single node using each parser are presented in the legend.

likely, it directly triggers Nougat rather than weighing its options with other moderate-cost parsers. Therefore, it does not invoke an LLM and skips stage CLS III. It employs pre-defined fastText (FT) word embeddings (Xu & Du, 2019).

The second variant, **AdaParse (LLM)**, implements the first classification stage, CLS I, to determine if the extracted text is worthy of being included in a batch and run through LLM inference. Once a batch of text items is assembled, this variant proceeds directly to the stage CLS III, which performs an LLM inference call to predict the most suitable parser for each text item. We employ SciBERT (Beltagy et al., 2019) for this task due to its high inference speed. Consequently, the single-node throughput is still $17\times$ higher than that achieved by solely relying on a state-of-the-art ViT-based parser (Nougat). We find that this use of LLM inference results in slightly lower throughput than the first variant—although still matching the performance of a text extraction tool such as pypdf—but offers higher accuracy and allows for better alignment with human preferences through DPO. The LLM-inferred labels determine if the extracted text is accepted as is or if the document (still in memory) is routed to a high-quality parser such as Nougat.

In essence, AdaParse is a meta-strategy that adaptively ensembles multiple parsers into a single, higher-accuracy system—loosely inspired by AdaBoost (Freund et al., 1996). We employ Parsl (Babuji et al., 2019), a pure-Python parallel scripting library, to orchestrate AdaParse’s data processing and model inference on the Polaris supercomputer.

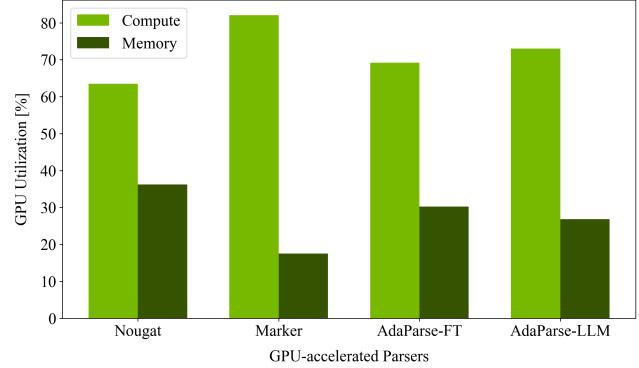


Figure 4. Utilization of the workload per GPU, as measured with the NVIDIA Nsight Systems profiler (Nsys).

5.2 Optimizing Parallel Execution on HPC Systems

Nougat serves as the high-quality parser in AdaParse. In the following, we restrict its usage to (at most) $\alpha = 5\%$ of the documents per node. Regardless, invoking Nougat requires loading a Swin-architecture-based Vision Transformer—which can take up to 15 seconds on an A100. Thus, we modify Parsl to allow Nougat to persist on each GPU beyond the task boundary. Since PyMuPDF, the best lightweight parser, runs exclusively on CPUs, there is effectively no competition with Nougat for GPUs, allowing for efficient resource sharing.

Nougat operates on a fixed image input size of $(H, W) = (896, 672)$, but allows control over how many pages are processed simultaneously. We find that a batch size of $B_p=10$ pages maximizes throughput without exceeding GPU memory capacity. Although Nougat’s Base model is relatively small (350M parameters), its memory footprint grows substantially when document pages are converted into image patches for the self-attention mechanism. By parsing pages individually at a fixed resolution—rather than entire documents—Nougat normalizes task size, resulting in more consistent execution times.

The performance analysis of the GPU-accelerated parsing methods was conducted using the NVIDIA Nsight Systems profiler (Nsys), and the results are presented in Figure 4.

6 EXPERIMENTAL METHODOLOGY

6.1 Hardware and Software Environment

All experiments were conducted on the Polaris system at the Argonne Leadership Computing Facility (ALCF). Polaris is an HPE Apollo Gen10+ system with 560 nodes interconnected by an HPE Slingshot-11 network with a Dragonfly topology. Each node consists of an AMD “Milan” processor with 32 cores and 512 GB of system memory, four 40 GB

NVIDIA A100 GPUs, and two Slingshot-11 25 GB/s network adapters. Each NVIDIA A100 GPU can achieve a peak of 19.5 TFLOPS in FP32 and 312 TFLOPS in FP16 and BF16. Polaris is supported by a Lustre file system, Eagle, residing on an HPE ClusterStor E1000 platform equipped with 100 PB of usable capacity across 8480 disk drives. This ClusterStor platform also provides 160 Object Storage Targets (OST) and 40 Metadata Targets (MT) with an aggregate data transfer rate of 650 GB/s. All experiment data is striped across 48 OSTs for optimal read and write bandwidth. As a current top 30 supercomputer, Polaris is representative of leadership class HPC systems.

We employ the Parsl workflow engine to orchestrate our PDF parsing effort efficiently. Parsl distributes tasks as pure functions—deterministic operations that do not modify shared program state—which poses challenges when the same ML model weights are needed across hundreds of workers pinned to GPUs. To mitigate this problem, we implement a warm-start mechanism for parsers requiring machine learning models. By loading the model weights once and persisting them across worker processes, we significantly reduce I/O overhead and initialization time for subsequent tasks. Furthermore, to decrease global I/O usage, we aggregate and chunk input files into a set of compressed ZIP archives and transfer them to node-local RAM storage. This strategy minimizes the frequent reading and writing of numerous small files to networked file systems, instead favoring larger, more efficient I/O operations suited to Lustre file systems. By processing data locally on each node, we enhance throughput and reduce the load on shared storage resources. Parsl dispatches tasks adaptively based on worker availability, ensuring efficient use of compute resources by dynamically balancing task distribution across nodes.

6.2 Document Selection and Preparation

We employ a diverse set of documents and formats, from both preprint servers and peer-reviewed publishers, for evaluating the parsers to ensure they can capture the diversity of scientific text. Diverse sources are important because different venues use different templates and represent different levels of polish in scientific works. The dataset includes documents sourced from ArXiv, BioRxiv, BMC, MDPI, MedRxiv, and Nature. The resulting collection spans eight domains (mathematics, biology, chemistry, physics, engineering, medicine, economics, and computer science) with 67 sub-categories ranging from acoustics to zoology. Including such a wide range of topics is critical to obtaining a comprehensive representation of different domain-specific features, such as extensive use of equations in mathematics and differing notations, conventions, citation schemes, and formatting used in various fields (Shah et al., 2021).

We focus on recent data that would not have been available

for ViT/OCR models to train on, in order to prevent data leakage from their training set into our test set. This choice presents a trade-off: it excludes older documents, which may contain metadata of varying quality for extraction tools like PyMuPDF.

To obtain groundtruth text for the benchmark, we parsed the HTML representation of a paper’s full text allowing us to obtain a sufficiently large number of documents for evaluation. Since HTML is straightforward to parse, it provides highly accurate groundtruth text.

Lastly, we perform page image and text layer manipulations (e.g., random scaling, artificial image imperfections, or modified metadata), as done in prior works (Zi, 2005; Groleau et al., 2023). This approach ensures that we evaluate the parsers as they would be applied “in the wild,” to obtain results that are representative of real-world performance.

6.3 User Preferences

Aligning accuracy with human preferences requires sampling those preferences. For this purpose, we launched a platform that allows domain experts to share their preferences on texts sourced from seven different parsers. The expert is presented with an image of a document page along with two parsed text outputs, and is prompted to either choose a preferred parse or indicate indifference if neither is preferred. The text formatting of the parser output was slightly modified to prevent bias (e.g., by including or removing hashtags that indicate markdown output from Nougat or Marker). Moreover, the selection of page and text pairs was non-adaptive to prevent user feedback bias (Mansoury et al., 2020). To ease users into this task, the website’s design emulates that of an OpenAI chatbot (Chiang et al., 2024). Moreover, users began annotating single paragraphs before moving on to entire document pages.

We engaged 23 scientists with expertise spanning mathematics, biology, physics, chemistry, medicine, engineering, and economics. We obtained 2794 preferences for 642 different document pages, which we partitioned into training, validation, and test subsets with sizes 712, 234, and 1848, respectively. The majority of the preferences were collected for the test subset to a) ensure a sufficient sample size to uphold the validity of the empirical results and b) present identical options to different users to assess consensus.

7 EVALUATION

After outlining how data was sourced and models configured, we present our results.

7.1 Alignment of Accuracy with User Preferences

We first need to assess if BLEU scores and similar metrics are good proxies for human preferences. To do this we study the outcomes of our user preference survey.

When aggregating over the entire dataset, users prefer Nougat the most (with a frequency of 57.1%) followed by Marker (49.1%) and PyMuPDF (48.6%). Throughput does not necessarily translate to user preferences. PyMuPDF, for example, offered a $2133 \times$ higher throughput while experiencing a BLEU score difference of 0.5%. However, users are not indifferent to parsers, as indicated by the low win frequency of 2.1% for pypdf. As these frequencies are determined by a binary tournament of different pairings of the seven parsers present in the study, percentages do not sum to 100%. Therefore, we report normalized win rates instead.

Users are highly willing to make their preferences known, doing so 91.3% of the time and while picking "neither" only in the remaining 8.7%. Moreover, participants have a high agreement in the choices they make. Among the 405 triplets of page document and two parser output texts shown to multiple users, participants made the same choice 82.2% of the time. This high consensus rate—achieved despite scientists' diverse disciplinary backgrounds—suggests a degree of objectivity in participants' preferences, underscoring their usefulness for model refinement. Importantly, these preferences are collected only once and used offline to adapt the model's weights via DPO during post-training, so that no further human input is required when the model is deployed for parsing.

A key result of this study is that the BLEU score, while indicative of user preference, is hardly predictive. The BLEU is highly correlated with the win rate (correlation $\hat{\rho} = 0.47$), which is statistically significant as $H_0 : \rho = 0$ is rejected with $p = 8.4^{-49}$. Yet the correlation is also far from 1, explaining only 47% of the variation in user choices. We view this result as justification that the BLEU score is a robust quality indicator of parser text output and a suitable target for LLM-finetuning, but also not completely predictive requiring the consideration of other measures of quality.

7.2 AdaParse Quality Assessment

Since AdaParse manages diverse parsers during its execution, it is important to probe the mechanism for parser selection and to gauge the improvement over using individual parsers. Because no single quality measure is completely predictive of user preferences, we consider a set of quality measures. The empirical investigation includes document coverage (as measured by the number of retrieved document pages), BLEU, ROUGE, and character-accuracy rate (CAR). It also includes two metrics we devised from the user preference study: *win rate* (WR) which measures how often a

parser was selected over the others for a given document and *accepted tokens* (AT) that tracks the relative frequency of tokens that exceed a critical BLEU threshold.

To evaluate AdaParse, we run it on a held-out test set of 1000 digitally born PDFs that were not used during the training. We report three rounds of metrics: the first with no changes to any layer of the PDFs, the second with augmentations applied only to the image layer, and the third with alterations applied only to the text layer.

We show in Table 1 the default quality on the test set. Marker has the highest coverage rate, but does not have the highest quality according to any other metric. Nougat has the highest win rate between parsers by a slim margin. AdaParse even with the requirement to allocate no more than 5% of the documents to its high-quality parser (Nougat), produces the best BLEU and ROUGE scores, and the second-best CAR. Additionally, AdaParse has the highest percentage of accepted tokens based on the user preference data at 76.9%. AdaParse can achieve better performance than any of its constituent parsers by delegating to the method that is most suitable for an individual document. While a parser like Nougat may perform best on average, it is not the best parser for each document allowing AdaParse to exceed it if it accurately infers a better parser-document matching.

The performance of AdaParse is based on the capability in predicting the BLEU score of PyMuPDF and Nougat-parsed text, with an $R^2 = 40.0\%$ and $R^2 = 46.5\%$, respectively. This is largely based on parameter-efficient finetuning through low-rank adaptation (LoRA) (Hu et al., 2021) and DPO on its weights in decoder mode. DPO post-training has been shown to improve performance in related downstream prediction tasks using relatively little preference data, even in high-performance computing (HPC) applications (Dharuman et al., 2024).

Table 1. Accuracy on born-digital PDFs: Document- (coverage rate), word- (BLEU, ROUGE), and character-level (CAR) accuracies. CAR = Character accuracy rate. WR = Win rate. AT = Accepted tokens. All %.

Parser	Coverage	BLEU	ROUGE	CAR	WR	AT
Marker	96.7	47.5	64.2	59.6	26.6	73.3
Nougat	93.0	48.1	66.5	65.8	27.9	69.8
PyMuPDF	91.3	51.9	67.3	67.0	24.4	76.7
pypdf	92.0	43.6	58.7	32.3	2.4	72.4
GROBID	81.0	26.5	52.4	54.8	—	20.6
Tesseract	91.3	48.8	64.2	67.8	18.7	72.5
AdaParse	91.5	52.1	67.6	67.1	25.5	76.9

Additionally, we test parsing performance under simulated image degradation to mimic low-quality scans. Low-quality scans are common in older academic and book datasets. We emulate this quality degradation with random rotations,

contrast adjustments, Gaussian blurring, and compression that are applied to a subset of 15% of documents, similar to the data augmentations used to train Nougat (Blecher et al., 2023). Note that these changes will not affect text extraction methods which is why we exclude them here. AdaParse shows favorable performance as it relies mostly on text extraction that is unaffected by these changes and Nougat that was trained to handle similar image augmentations. The only statistically meaningfully affected metric is the win rate—in part because AdaParse is artificially limited to selecting an image parser when its quality is higher. However, it is important to note that this does not translate to a lower token acceptance rate, as many documents are still parsed above the acceptance threshold.

Table 2. Accuracy on simulated scanned PDFs: Document- (coverage rate), word- (BLEU, ROUGE), and character-level (CAR) accuracies. All %.

Parser	Coverage	BLEU	ROUGE	CAR	WR	AT
Marker	96.5	46.6	62.9	60.5	28.0	70.1
Nougat	91.9	45.1	63.1	63.4	27.2	63.5
Tesseract	90.0	44.0	58.2	65.2	12.8	59.0
AdaParse	92.8	52.0	67.5	67.0	18.4	77.0

Finally, we investigate the perturbation of the text layer. 15% of the embedded text layers are replaced with the output of common tools (Tesseract or GROBID), explaining their exclusion in the table. This configuration tests the ability to determine when a higher-quality parse is needed because of degraded text. Few documents parsed by Nougat are sufficient to give AdaParse an edge in this setting. Given that we still limit AdaParse’s choice of image parsers to at most 5% of the dataset compared to the 15% where the text layer is removed, it is unsurprising that quality degrades commensurate with the text parsers that are being used for the bulk of the parsing efforts. Regardless, AdaParse correctly delegates sufficiently many of those documents to other parsers which is why quality remains higher than using text extraction-based parsers alone.

Overall, AdaParse offers robust performance across data regimes.

Table 3. Accuracy on PDFs with simulated OCR-degraded text layers: Document- (coverage rate), word- (BLEU, ROUGE), and character-level (CAR) accuracies. All %.

Parser	Coverage	BLEU	ROUGE	CAR	WR	AT
PyMuPDF	90.8	42.0	55.6	56.5	13.1	58.8
pypdf	91.2	35.6	48.9	29.8	1.2	56.9
AdaParse	91.2	42.4	55.9	56.7	12.0	59.5

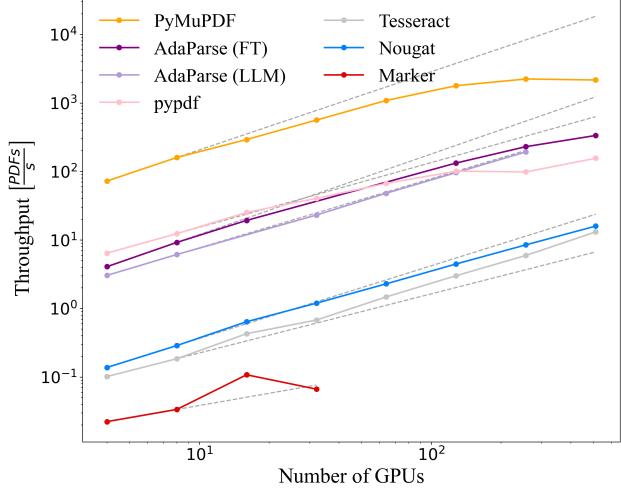


Figure 5. Scalability of the seven parsers.

7.3 Throughput Scalability

In addition to quality, throughput is the key evaluation criterion for large-scale parsing techniques. If you cannot perform a parse with a given parser due to insufficient resources, the quality of that parser becomes moot.

We evaluate the performance of each method on Polaris using between 1 and 128 nodes and present the results in Figure 5. We see that text extraction-based methods such as PyMuPDF are the fastest, processing up to ≈ 315 PDF/second at scale, while methods like Marker fail to scale beyond 10 nodes, producing on average only 0.1 PDF/second. Nougat offers slightly better throughput with ≈ 8 PDF/second on 128 nodes. AdaParse (FT) exhibits intermediate performance with ≈ 78 PDF/second. Most methods scale roughly linearly in the number of nodes. Notable exceptions include PyMuPDF and pypdf, which initially scale linearly but plateau at around 128 and 100 nodes, respectively. In the case of PyMuPDF, plateauing occurs because extraction is sufficiently fast that contention for file system resources begins to be the bottleneck, limiting scalability.

8 CONCLUSION

A key step in extracting knowledge from scientific documents is to improve the quality of PDF document parsing. This step has long been a bottleneck for building AI foundation models for science. While tools developed for Internet-scale data are widely adopted for training advanced AI models, training on text encoded within scientific literature data remains an open challenge. We have presented AdaParse as a practical approach to address this challenge—mainly by incorporating a data-driven strategy to distribute

each parsing task to the most appropriate PDF parser tool, in a portable, yet reusable manner. We also developed direct preference optimization for rating PDF parser quality, allowing selections to be aligned with human judgment. We show that the resulting solution is able to leverage existing PDF parsing tools for large-scale campaigns that make effective use of high-performance computing infrastructure.

ACKNOWLEDGMENTS

This research used resources of the Argonne Leadership Computing Facility, a U.S. Department of Energy (DOE) Office of Science user facility at Argonne National Laboratory (ANL) and is based on research supported by the DOE Office of Science–Advanced Scientific Computing Research Program and by Laboratory Directed Research and Development (LDRD) funding from ANL, provided by the Director, DOE Office of Science, both under Contract No. DE-AC02-06CH11357.

REFERENCES

- Grobid. <https://github.com/kermitt2/grobid>, 2008–2025.
- Adhikari, N. S. and Agarwal, S. A comparative study of pdf parsing tools across diverse document categories. *arXiv preprint arXiv:2410.09871*, 2024.
- Artifex Software. MuPDF. URL <https://mupdf.com/>. Accessed: Oct 2024.
- Babuji, Y., Woodard, A., Li, Z., Clifford, B., Kumar, R., Lacinski, L., Chard, R., Wozniak, J., Foster, I., Wilde, M., Katz, D., and Chard, K. Parsl: Pervasive parallel programming in Python. In *ACM International Symposium on High-Performance Parallel and Distributed Computing*, 2019.
- Bast, H. and Korzen, C. A benchmark and evaluation for text extraction from pdf. In *2017 ACM/IEEE joint conference on digital libraries (JCDL)*, pp. 1–10. IEEE, 2017.
- Beltagy, I., Lo, K., and Cohan, A. Scibert: A pre-trained language model for scientific text. *arXiv preprint arXiv:1903.10676*, 2019.
- Blecher, L., Cucurull, G., Scialom, T., and Stojnic, R. Nouagat: Neural optical understanding for academic documents. *arXiv preprint arXiv:2308.13418*, 2023.
- Bulan, O., Kozitsky, V., Ramesh, P., and Shreve, M. Segmentation-and annotation-free license plate recognition with deep localization and failure identification. *IEEE Transactions on Intelligent Transportation Systems*, 18(9):2351–2363, 2017.
- Chiang, W.-L., Zheng, L., Sheng, Y., Angelopoulos, A. N., Li, T., Li, D., Zhang, H., Zhu, B., Jordan, M., Gonzalez, J. E., and Stoica, I. Chatbot Arena: An open platform for evaluating LLMs by human preference. *arXiv preprint arXiv:2403.04132*, 2024.
- Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N. M., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., García, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A. M., Pillai, T. S., Pelлат, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Díaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K., Eck, D., Dean, J., Petrov, S., and Fiedel, N. PaLM: Scaling

- language modeling with pathways. *J. Mach. Learn. Res.*, April 2022.
- Cohan, A., Feldman, S., Beltagy, I., Downey, D., and Weld, D. S. Specter: Document-level representation learning using citation-informed transformers. *arXiv preprint arXiv:2004.07180*, 2020.
- Corrêa, A. S. and Zander, P.-O. Unleashing tabular content to open data: A survey on PDF table extraction methods and tools. In *18th Annual International Conference on Digital Government Research*, pp. 54–63, 2017.
- Coulon, R., Toro, F. G., and Michotte, C. Machine-readable data and metadata of international key comparisons in radionuclide metrology. *Measurement Science and Technology*, 34(7):074009, 2023.
- Dharuman, G., Hippe, K., Brace, A., Foreman, S., Hatanpää, V., Sastry, V. K., Zheng, H., Ward, L., Muralidharan, S., Vasan, A., et al. Mprot-dpo: Breaking the exaflops barrier for multimodal protein design workflows with direct preference optimization. In *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–13. IEEE, 2024.
- Du, Y., Li, C., Guo, R., Yin, X., Liu, W., Zhou, J., Bai, Y., Yu, Z., Yang, Y., Dang, Q., et al. PP-OCR: A practical ultra lightweight OCR system. *arXiv preprint arXiv:2009.09941*, 2020.
- Elazar, Y., Bhagia, A., Magnusson, I., Ravichander, A., Schwenk, D., Suhr, A., Walsh, P., Groeneveld, D., Soldaini, L., Singh, S., et al. What’s in my big data? *arXiv preprint arXiv:2310.20707*, 2023.
- Fenniak, M., Stamy, M., pubpub zz, Thoma, M., Peveler, M., exiledkingcc, and pypdf Contributors. The pypdf library, 2024. URL <https://pypi.org/project/pypdf/>.
- Freund, Y., Schapire, R. E., et al. Experiments with a new boosting algorithm. In *icml*, volume 96, pp. 148–156. Citeseer, 1996.
- Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., Presser, S., and Leahy, C. The Pile: An 800GB dataset of diverse text for language modeling, December 2020. URL <http://arxiv.org/abs/2101.00027>. arXiv:2101.00027 [cs].
- Graham, Y. Re-evaluating automatic summarization with bleu and 192 shades of rouge. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pp. 128–137, 2015.
- Groleau, A., Chee, K. W., Larson, S., Maini, S., and Boarman, J. Augraphy: A data augmentation library for document images. In *International Conference on Document Analysis and Recognition*, pp. 384–401. Springer, 2023.
- Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.
- Jaume, G., Ekenel, H. K., and Thiran, J.-P. FUNSD: A dataset for form understanding in noisy scanned documents. In *International Conference on Document Analysis and Recognition Workshops*, volume 2, pp. 1–6. IEEE, 2019.
- Jimeno Yepes, A., Zhong, P., and Burdick, D. Icdar 2021 competition on scientific literature parsing. In *16th International Conference on Document Analysis and Recognition*, pp. 605–617. Springer, 2021.
- Kim, G., Hong, T., Yim, M., Nam, J., Park, J., Yim, J., Hwang, W., Yun, S., Han, D., and Park, S. OCR-free document understanding transformer. In *European Conference on Computer Vision*, pp. 498–517. Springer, 2022.
- Kurabayashi, M. and Wong, K. StealthPDF: Data hiding method for PDF file with no visual degradation. *Journal of Information Security and Applications*, 61:102875, 2021.
- Lab, D. Marker - data lab. <https://www.datalab.to/marker>, 2024. Accessed: 2024-10-29.
- Levenshtein, V. Binary codes capable of correcting deletions, insertions, and reversals. *Proceedings of the Soviet physics doklady*, 1966.
- Li, M., Lv, T., Chen, J., Cui, L., Lu, Y., Florencio, D., Zhang, C., Li, Z., and Wei, F. TrOCR: Transformer-based optical character recognition with pre-trained models. In *AAAI Conference on Artificial Intelligence*, volume 37, pp. 13094–13102, 2023.
- Li, S., Huang, J., Zhuang, J., Shi, Y., Cai, X., Xu, M., Wang, X., Zhang, L., Ke, G., and Cai, H. Scilitlm: How to adapt llms for scientific literature understanding. *arXiv preprint arXiv:2408.15545*, 2024.
- Lin, C.-Y. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, 2004.
- Lo, K., Wang, L. L., Neumann, M., Kinney, R., and Weld, D. S. S2ORC: The Semantic Scholar open research corpus. *arXiv preprint arXiv:1911.02782*, 2019.

- Lopez, P. Grobid: Combining automatic bibliographic data recognition and term extraction for scholarship publications. In *International conference on theory and practice of digital libraries*, pp. 473–474. Springer, 2009.
- Mansouri, M., Abdollahpouri, H., Pechenizkiy, M., Mobasher, B., and Burke, R. Feedback loop and bias amplification in recommender systems. In *29th ACM International Conference on Information & Knowledge Management*, pp. 2145–2148, 2020.
- Mujumdar, S., Gupta, N., Jain, A., and Burdick, D. Simultaneous optimisation of image quality improvement and text content extraction from scanned documents. In *International Conference on Document Analysis and Recognition*, pp. 1169–1174. IEEE, 2019.
- Nerbonne, J., Heeringa, W., and Kleiweg, P. Edit distance and dialect proximity. *Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison*, 15, 1999.
- Neudecker, C., Baierer, K., Gerber, M., Clausner, C., Antonacopoulos, A., and Pletschacher, S. A survey of OCR evaluation tools and metrics. In *6th International Workshop on Historical Document Imaging and Processing*, pp. 13–18, 2021.
- Nguyen, T. T. H., Jatowt, A., Coustaty, M., and Doucet, A. Survey of post-OCR processing approaches. *ACM Computing Surveys (CSUR)*, 54(6):1–37, 2021.
- Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. BLEU: A method for automatic evaluation of machine translation. In *40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318, 2002.
- Paudel, P., Khadka, S., Shah, R., et al. Optimizing Nepali PDF extraction: A comparative study of parser and OCR technologies. *arXiv preprint arXiv:2407.04577*, 2024.
- Post, M. A call for clarity in reporting bleu scores. *arXiv preprint arXiv:1804.08771*, 2018.
- Quattrini, F., Zaccagnino, C., Cascianelli, S., Righi, L., and Cucchiara, R. μ gat: Improving single-page document parsing by providing multi-page context. *arXiv preprint arXiv:2408.15646*, 2024.
- Ravi, S., Knight, K., and Soricut, R. Automatic prediction of parser accuracy. In *Conference on Empirical Methods in Natural Language Processing*, pp. 887–896, 2008.
- Reiter, E. A structured review of the validity of bleu. *Computational Linguistics*, 44(3):393–401, 2018.
- Saiga, H., Nakamura, Y., Kitamura, Y., and Morita, T. An OCR system for business cards. In *2nd International Conference on Document Analysis and Recognition*, pp. 802–805. IEEE, 1993.
- Shaffi, N. and Hajamohideen, F. uTHCD: A new benchmarking for Tamil handwritten OCR. *IEEE Access*, 9: 101469–101493, 2021.
- Shah, A. K., Dey, A., and Zanibbi, R. A math formula extraction and evaluation framework for pdf documents. In *Document Analysis and Recognition–ICDAR 2021: 16th International Conference, Lausanne, Switzerland, September 5–10, 2021, Proceedings, Part II 16*, pp. 19–34. Springer, 2021.
- Singh, P., Tapaswi, S., and Gupta, S. Malware detection in PDF and Office documents: A survey. *Information Security Journal: A Global Perspective*, 29(3):134–153, 2020.
- Smith, R. An overview of the Tesseract OCR engine. In *9th International Conference on Document Analysis and Recognition*, volume 2, pp. 629–633. IEEE, 2007.
- Soldaini, L., Kinney, R., Bhagia, A., Schwenk, D., Atkinson, D., Author, R., Beglin, B., Chandu, K., Dumas, J., Elazar, Y., Hofmann, V., Jha, A. H., Kumar, S., Lucy, L., Lyu, X., Lambert, N., Magnusson, I., Morrison, J., Muennighoff, N., Naik, A., Nam, C., Peters, M. E., Ravichander, A., Richardson, K., Shen, Z., Strubell, E., Subramani, N., Tafjord, O., Walsh, P., Zettlemoyer, L., Smith, N. A., Hajishirzi, H., Beltagy, I., Groeneveld, D., Dodge, J., and Lo, K. Dolma: An open corpus of three trillion tokens for language model pretraining research, June 2024. URL <http://arxiv.org/abs/2402.00159>. arXiv:2402.00159 [cs].
- Sorscher, B., Geirhos, R., Shekhar, S., Ganguli, S., and Morcos, A. S. Beyond neural scaling laws: Beating power law scaling via data pruning, April 2023. URL <http://arxiv.org/abs/2206.14486>. arXiv:2206.14486 [cs].
- Steedman, M., Osborne, M., Sarkar, A., Clark, S., Hwa, R., Hockenmaier, J., Ruhlen, P., Baker, S., and Crim, J. Bootstrapping statistical parsers from small datasets. In *10th Conference on European Chapter of the Association for Computational Linguistics–Volume 1*, pp. 331–338. Association for Computational Linguistics, 2003.
- Taylor, R., Kardas, M., Cucurull, G., Scialom, T., Hartshorn, A., Saravia, E., Poulton, A., Kerkez, V., and Stojnic, R. Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085*, 2022.
- Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., and Zhou, M. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers.

Advances in neural information processing systems, 33: 5776–5788, 2020.

Xu, J. and Du, Q. A deep investigation into fastText.

In *IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pp. 1714–1719. IEEE, 2019.

Zhu, W., Sokhadan, N., Yang, G., Martin, S., and Sathyanarayana, S. DocBed: A multi-stage OCR solution for documents with complex layouts. In *AAAI Conference on Artificial Intelligence*, volume 36, pp. 12643–12649, 2022.

Zi, G. *Groundtruth generation and document image degradation*. University of Maryland, College Park, 2005.

Zuidema, W. Parsimonious data-oriented parsing. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 551–560, 2007.

A THE DPO FORMALISM

We denote the parsed text of document i by parser j as $x_i^j = \phi_j(d_i)$ with accuracy (e.g., BLEU score) y_i^j . Hence, the dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ represents the (parsed) text inputs with \mathbb{R}^m -valued responses (i.e., a document-wise accuracy vector). We post-train a model to predict the accuracies of all parsers given the default parser’s text ϕ_1^1 in three steps. First, supervised fine-tuning yields the estimate $\hat{\theta}_1$ through minimization of the ℓ_2 loss

$$\mathcal{L}_{\text{REG}}(\theta) = \mathbb{E}_{\mathcal{D}} [\|\pi_\theta(x^1) - y\|_2^2].$$

Second, $\pi_{\hat{\theta}_1}$ is augmented into an encoder-decoder model g_φ , with $\text{Enc}_{\varphi_e}(x) = h$ and $\text{Dec}_{\varphi_d}(h) = z$, where $\varphi = (\varphi_e, \varphi_d)$ and $\varphi_e := \hat{\theta}_1$ initially. We utilize a preference dataset $\mathcal{D}_{\text{pref}} = \{(x_j^{k_1,+}, x_j^{k_2,-})\}_{j=1}^M$ of text pairs obtained through different parsers ϕ_{k_1} and ϕ_{k_2} where the former is preferred by the user. Minimizing

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E}_{\mathcal{D}_{\text{pref}}} \left[\log \sigma \left(\theta \log \frac{g_\varphi(x^+)}{g_\varphi^{\text{ref}}(x^+)} - \log \frac{g_\varphi(x^-)}{g_\varphi^{\text{ref}}(x^-)} \right) \right]$$

upon convergence yields the estimate $\hat{\theta}_2 = \hat{\varphi}_e$. Finally, the updated encoder is fine-tuned on \mathcal{D} with a lowered learning rate to obtain $\hat{\theta}_3$ which produces the final model.

In our setting, the regression dataset contains $N=29,200$ pairs, each consisting of a single document text and its associated BLEU score. The output dimension is $m=6$, since we predict the accuracy for each parser. The preference dataset contains $M=712$ pairs. We found it advantageous in step 1 to predict pagewise accuracy (i.e., predict the accuracy of the given page’s parsed text), while the regression data in the third step are used to infer document-level accuracy based on the first page’s text, as processed by AdaParse.

B QUANTIFICATION OF THE DPO IMPACT

We quantify the benefit of direct preference optimization (DPO) by evaluating a range of prediction models. As a baseline, we apply support vector classification (SVC) to metadata features (e.g., publisher, year of publication, PDF format, and producer). LLM-based prediction of the document text is performed with SciBERT, BERT, MiniLM, and SPECTER (Cohan et al., 2020; Wang et al., 2020). The metrics of the reference models (BLEU-maximal/minimal and random selection) are provided for context.

Given the six parsers, predicting the optimal choice for any PDF is challenging. The assignment of the BLEU-maximal parser to each document yields a BLEU score of 56.8%. Although metadata-driven classification delivers (mostly) favorable results, text-driven regression with LLMs outperforms them across all metrics. Post-training through DPO further boosts BLEU, CAR, and win rate. Transformer-based models pre-trained on extensive scientific corpora,

Table 4. Evaluation of various prediction models across different features. Word-level (BLEU, ROUGE) and character level accuracy (CAR) accuracies. WR=Win rate. All %.

Features (Model)	BLEU	ROUGE	CAR	WR	ACC
CLS III: Document Text					
Text (SciBERT + DPO)	52.7	69.4	68.0	31.4	36.7
Text (SciBERT)	51.6	69.5	66.9	25.0	48.3
Text (BERT)	49.7	66.0	63.4	24.8	40.0
CLS II: Metadata and Title Text					
Title + Metadata (SPECTER)	47.9	64.5	62.9	25.2	18.1
Title (SPECTER)	46.4	63.3	61.8	26.2	15.2
Title + Metadata (MiniLM-L6)	44.7	62.2	60.4	28.4	10.1
CLS I: Metadata					
Format + Producer (SVC)	47.7	64.0	60.2	28.5	14.6
Format (SVC)	47.5	64.1	60.7	29.5	16.6
Year + Producer (SVC)	47.3	63.7	60.1	28.8	14.8
Publisher + (Sub-)category (SVC)	46.4	63.7	60.9	21.7	14.8
(Sub-)category (SVC)	43.6	63.5	62.5	24.9	12.9
Reference					
BLEU-maximal selection	56.8	72.3	70.4	26.5	100.0
Random selection	44.0	61.7	57.4	20.5	16.7
BLEU-minimal selection	21.5	44.2	44.6	18.1	0.0

such as SciBERT and SPECTER, outperform models trained on conventional web-scale data like BERT and MiniLM-v6. AdaParse (LLM) leverages SciBERT with DPO post-training for parser selection.

C SOLVING THE OPTIMIZATION PROBLEM

For scalability reasons, AdaParse limits itself to two parsers: PyMuPDF and Nougat. The problem turns to picking either ϕ_{Nougat} or ϕ_{PyMuPDF} for any document d_i . The average computational cost of a parser can be determined from our scaling experiments and is documented in the legend of Figure 3. They are denoted by $\mathcal{T}_{\text{Nougat}}^{\text{avg}}$ and $\mathcal{T}_{\text{PyMuPDF}}^{\text{avg}}$. The parameter $\alpha \in [0, 1]$ limits the fraction of documents parsed with Nougat. The constraint

$$\sum_{i=1}^n \mathcal{T}(\phi_{j_i}, d_i) \approx \alpha n \left(\mathcal{T}_{\text{Nougat}}^{\text{avg}} - \mathcal{T}_{\text{PyMuPDF}}^{\text{avg}} \right) + n \mathcal{T}_{\text{PyMuPDF}}^{\text{avg}} \leq \bar{\mathcal{T}}$$

is (approximately) satisfied for any

$$\alpha \leq \frac{\bar{\mathcal{T}} - n \mathcal{T}_{\text{PyMuPDF}}^{\text{avg}}}{n \left(\mathcal{T}_{\text{Nougat}}^{\text{avg}} - \mathcal{T}_{\text{PyMuPDF}}^{\text{avg}} \right)}.$$

The objective function is now maximized when sorting the documents (by expected accuracy improvement of Nougat over PyMuPDF) and allowing the first $\lfloor \alpha n \rfloor$ documents to be parsed by Nougat. AdaParse conducts this on a per-batch basis to further increase throughput (i.e. for a batch of size k at most $\lfloor \alpha k \rfloor$ documents will be parsed by Nougat). While this per-batch approach may yield a suboptimal solution, the optimality gap is negligible as the batch size is large (e.g. $k=256$ in our case).