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

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

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, 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.

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

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 $\mathcal{T}_{\text{Nougat}}^{\text{avg}}$. The parameter $\alpha \in [0,1]$ limits the fraction of documents parsed with Nougat. The constraint

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

is (approximately) satisfied for any

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

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).

D Artifact Appendix

D.1 Artifact check-list (meta-information)

- **Algorithm:** AdaParse
- **Program:** Python
- **Compilation:** Not applicable (pure Python)
- **Transformations:** YAML configuration for workflow
- **Binary:** N/A
- **Data set:** Arbitrary (zipped) PDFs provided by the user
- **Run-time environment:** Python 3.12, conda environment
- **Hardware:** GPU required
- **Execution:** Command-line interface
- **Metrics:** Throughput, accuracy of PDF parsing, and quality of parser outputs
- **Output:** N/A (functionality only)
- **Experiments:** Demonstrate functionality
- **How much disk space required (approximately)?:** Less than a few GBs for a small dataset
- **How much time is needed to prepare workflow (approximately)?:** 15–30 minutes for setup
- **How much time is needed to complete experiments (approximately)?:** 15 minutes per job
- **Publicly available?:** https://github.com/7shoe/AdaParse
- **Code licenses (if publicly available)?:** MIT License
- **Workflow framework used?:** Custom Python scripts integrated with PBS scheduling

D.2 Installation

The steps below enable any of the parsers:

```
conda create -n adaparse python=3.12 -y
conda activate adaparse

# git repo (machine-agnostic)
git clone git@github.com:7shoe/AdaParse.git
cd AdaParse
pip install --upgrade pip setuptools wheel
pip install -e .
```

D.3 Artifact Configuration File

The artifact requires adoption of the file in `https://github.com/7shoe/AdaParse/blob/main/examples/pymupdf/pymupdf_test.yaml`. In particular, ensure the paths `pdf_dir` and `out_dir` are valid. For example, the YAML file should include:

```
# The directory containing the pdfs
pdf_dir: {path_to_(zipped)_pdfs}

# The directory for converted texts
out_dir: {output_directory}

# The settings for the pdf parser
parser_settings:
  # The name of the parser to use
  name: pymupdf
```