Paper OCR to Markdown2:The Missing Piece - Post-Processing Algorithms

type	Post
status	Invisible
date	@2025/10/09
slug	How-to-Convert-Paper-OCR-to-Markdown-2
tags	How-To
category	Guide

Problem Solved!

After a week of struggling with PDF-to-Markdown conversion (documented in previous post), I finally found the solution. Turns out, **it wasn't a model problem at all** — I was missing a crucial piece of the puzzle: **post-processing algorithms**.

🔍 What I Was Missing

My initial approach focused heavily on finding the perfect OCR model and layout detection tool. I tried:

- DocLayout-YOLO + various OCR engines
- Meta's Nougat (great results, but image positioning was off)
- LayoutParser
- Creative workarounds like masking images with text markers

But the core issue remained: **images appeared in wrong positions**, making the output unusable for actual reading.

💡 The Breakthrough: MinerU

Then I discovered MinerU, an open-source project by OpenDataLab. What makes it different isn't just the models — it's the **complete pipeline with sophisticated post-processing algorithms**.

Key Components:

- **Layout Detection**: LayoutLMv3_ft
- **Formula Detection**: YOLOv8_ft
- **Formula Recognition**: UniMERNet
- **OCR**: PaddleOCR
- **Table Recognition**: RapidTable

But more importantly...

🧩 The Magic Behind the Scenes: Post-Processing Algorithms

This is where MinerU truly shines. According to the documentation, it implements several crucial algorithms:

1. Reading Order Reconstruction

[To be researched: detailed algorithm implementation]

This algorithm reorganizes detected elements (text blocks, images, tables) into the correct reading sequence, handling complex layouts like:

- Multi-column papers
- Figures spanning multiple columns
- Footnotes and references
- *[Details on spatial relationship analysis - TBD]*

2. Intelligent Image Positioning & Referencing

[To be researched: specific positioning strategy]

The algorithm doesn't just detect images — it:

- Analyzes contextual references in text
- Places images near their first mention
- *[How it handles figure citations - TBD]*
- *[Coordinate system and placement rules - TBD]*

3. Smart Text Merging

[To be researched: merging criteria and rules]

Combines fragmented OCR text outputs intelligently:

- Proximity-based merging
- Semantic coherence checking
- *[Language model integration details - TBD]*

4. Layout Analysis & Semantic Understanding

- Identifies document structure (titles, paragraphs, captions)
- Distinguishes between main content and auxiliary elements
- *[Classification algorithm details - TBD]*

✅ Results

After implementing MinerU with its full pipeline:

- ✅ Images appear exactly where they should
- ✅ Formulas properly converted to LaTeX
- ✅ Tables extracted accurately
- ✅ Reading order preserved correctly
- ✅ Output is actually readable without manual fixes!

🔬 Next Steps

I plan to dive deeper into MinerU's post-processing algorithms to understand:

1. How the reading order reconstruction actually works
2. The specific rules for image positioning
3. The text merging strategy
4. Whether I can adapt/improve these algorithms for specific use cases

Stay tuned for a detailed technical breakdown!

📚 Resources

- MinerU GitHub
- MinerU Website
- Related Project: dots.ocr

TL;DR: The solution to accurate PDF-to-Markdown conversion wasn't better models, but better **post-processing algorithms** that handle layout reconstruction and image positioning. MinerU solved this perfectly.