iPhylo

Document layout analysis

Roderic Page

Published August 31, 2023

Citation

Page, R. (2023). Document layout analysis. *Iphylo*. https://doi.org/10.59350/z574z-dcw92

Keywords

ABBYY, CRF, DjVu, Document Layout, HOCR

How to cite: **Page, R. (2023). Document layout analysis. https://doi.org/10.59350/z574z-dcw92**

Some notes to self on document layout analysis.

I'm revisiting the problem of taking a PDF or a scanned document and determining its structure (for example, where is the title, abstract, bibliography, where are the figures and their captions, etc.). There are lots of papers on this topic, and lots of tools. I want something that I can use to process both born-digital PDFs and scanned documents, such as the ABBYY, DjVu and hOCR files on the Internet Archive. PDFs remain the dominant vehicle for publishing taxonomic papers, and aren't going away any time soon (see Pettifer et al. for a nuanced discussion of PDFs).

There are at least three approaches to document layout analysis.

Rule-based

The simplest approach is to come up rules, such as "if the text is large and it's on the first page, it's the title of the article". Examples of more sophisticated rules are given in Klampfl et al., Ramakrishnan et al., and Lin. Rule-based methods can get you a long way, as shown by projects such as Plazi. But there are always exceptions to rules, and so the rules need constant tweaking. At some point it makes sense to consider probabilistic methods that allow for uncertainty, and which can also "learn".

Large language models (LLMs)

At the other extreme are Large language models (LLMs), which have got a lot of publicity lately. There are a number of tools that use LLMs to help extract information from documents, such as LayoutLM (Xu et al.), Layout Parser, and VILA (Shen et al.). These approaches encode information about a document (in some case including the (*x*,*y*) coordinates of individual words on a page) and try and infer which category each word (or block of text) belongs to. These methods are typically coded in Python, and come with various tools to display regions on pages. I've had variable success getting these tools to work (I am new to Python, and am also working on a recent Mac which is not the most widely used hardware for machine learning). I have got other ML tools to work, such as an Inception-based model to classify images (see Adventures in machine learning: iNaturalist, DNA barcodes, and Lepidoptera), but I've not succeeded in training these models. There are obscure Python error messages, some involving Hugging Face, and eventually my patience wore out.

Another aspect of these methods is that they often package everything together, such that they take a PDF, use OCR or ML methods such as Detectron to locate blocks, then encode the results and feed them to a model. This is great, but I don't necessarily want the whole package, I want

just some parts of it. Nor does the prospect of lengthy training appeal (even if I could get it to work properly).

The approach that appealed the most is VILA, which doesn't use (*x*,*y*) coordinates directly but instead encodes information about "blocks" into text extracted from a PDF, then uses an LLM to infer document structure. There is a simple demo at Hugging Face. After some experimentation with the code, I've ended up using the way VILA represents a document (a JSON file with a series of pages, each with lists of words, their positions, and information on lines, blocks, etc.) as the format for my experiments. If nothing else this means that if I go back to trying to train these models I will have data already prepared in an appropriate format. I've also decided to follow VILA's scheme for labelling words and blocks in a document:

- Title
- Author
- Abstract
- Keywords
- Section
- Paragraph
- List
- Bibliography
- Equation
- Algorithm
- Figure
- Table
- Caption
- Header
- Footer
- Footnote

I've tweaked this slightly by adding two additional tags
from VILA's Labeling Category Reference, the "semantic" tags "Affiliation" and "Venue". This helps separate information on author names ("Author") from their affiliations, which can appear in very different positions to the author's names. "Venue" is useful to label things such as a banner at the top of an article where the publisher display the name of the journal, etc.

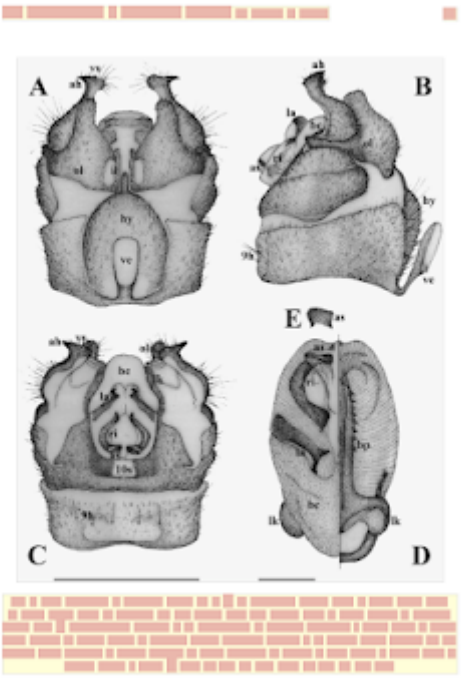
Conditional random fields

In between masses of regular expressions and large language models are approaches such as Conditional random fields (CRFs), which I've used before to parse citations (see Citation parsing tool released). Well known tools such as GROBID use this approach.

CRFs are fast, and somewhat comprehensible. But it does require Feature engineering, that is, you need to come up with features of the data to help train the model (for the systematists among you, this is very like coming up with characters for a bunch of taxa). This is were you can reuse the rules developed in a rules-based approach, but instead of having the rules make

decisions (e.g., “big text = Title”), you just a rule that detects whether text is big or not, and the model combined with training data then figures out if and when big text means “Title”. So you end up spending time trying to figure out how to represent document structure, and what features help the model get the right answer. For example, methods such as Lin’s for detecting whether there are recurring elements in a document are great source of features to help recognise headers and footers. CRFs also make it straightforward to include dependencies (the “conditional” in the name). For example, a bibliography in a paper can be recognised not just by a line having a year in it (e.g., “2020”), but there being nearby lines that also have years in them. This helps us avoid labelling isolated lines with years as “Bibliography” when they are simply text in a paragraph that mentions a year.

Compared to LLMs this a lot of work. In principle with an LLM you “just” take a lot of training data (e.g., text and location on a page) and let the model to the hard work of figuring out which bit of the document corresponds to which category (e.g., title, abstract, paragraph, bibliography). The underlying model has already been trained on (potentially) vast amounts of text (and sometimes also word coordinates). But on the plus side, training CRFs is very quick, and hence you can experiment with adding or removing features, adding training data, etc. For example, I’ve started training with about ten (10) documents, training takes seconds, and I’ve got serviceable results.



Lots of room for improvement, but there’s a constant feedback loop of seeing improvements, and thinking about how to tweak the features. It also encourages me to think about what went wrong.

Problems with PDF parsing

To process PDFs, especially "born digital" PDFs I rely on pdf2xml, originally written by Hervé Déjean (Xerox Research Centre Europe). It works really well, but I've encountered a few issues. Some can be fixed by adding more fonts to my laptop (from XpdfReader), but others are more subtle.

The algorithm used to assign words to "blocks" (e.g., paragraphs) seems to struggle with superscripts (e.g., [1]), which often end up being treated as separate blocks. This breaks up lines of text, and also makes it harder to accurately label parts of the document such as "Author" or "Affiliation".

Figures can also be problematic. Many are simply bitmaps embedded in a PDF and can be easily extracted, but sometimes labelling on those bitmaps, or indeed big chunks of vector diagrams are treated as text, so we end up with story text blocks in odd positions. I need to spend a little time thinking about this as well. I also need to understand the "vet" format pdftoxml extracts from PDFs.

PDFs also have all sorts of quirks, such as publishers slapping cover pages on the front, which make feature engineering hard (the biggest text might now be not be the title but some cruff from the publisher). Sometimes there are clues in the PDF that it has been moodier.! For example, ResearchGate inserts a "rgid" tag in the PDF when it adds a cover page.

Yes but why?

So, why I am doing this? Why battle with the much maligned PDF format. It's because a huge chunk of taxonomic and other information is locked up in PDFs, and I'd like a simpler, scalable, way to extract some of that. Plazi is obviously the leader in this are in terms of the amount of information they have extracted, but their approach is labour-intensive. I want something that is essentially automatic, that can be trained to handle the idiosyncracities of the taxonomic literature, and can be applied to both born digital PDFs and OCR from scans in the Biodiversity Heritage Library and elsewhere. Even if we could simply extract bibliographic information (to flesh out the citation graph) and the figures, that would be progress.

References

Déjean H, Meunier J-L (2006) A System for Converting PDF Documents into Structured XML Format. In: Bunke H, Spitz AL (eds) Document Analysis Systems VII. Springer, Berlin, Heidelberg, pp 129–140 https://doi.org/10.1007/11669487_12

Klampfl S, Granitzer M, Jack K, Kern R (2014) Unsupervised document structure analysis of digital scientific articles. Int J Digit Libr 14(3):83–99. https://doi.org/10.1007/s00799-014-0115-1

Lin X (2003) Header and footer extraction by page association. In: Document Recognition and Retrieval X. SPIE, pp 164–171 https://doi.org/10.1117/12.472833

Pettifer S, McDERMOTT P, Marsh J, Thorne D, Villeger A, Attwood TK (2011) Ceci n'est pas un hamburger: modelling and representing the scholarly article. Learned Publishing 24(3):207–220. https://doi.org/10.1087/20110309

Ramakrishnan C, Patnia A, Hovy E, Burns GA (2012) Layout-aware text extraction from full-text PDF of scientific articles. Source Code for Biology and Medicine 7(1):7. https://doi.org/10.1186/1751-0473-7-7

Shen Z, Lo K, Wang LL, Kuehl B, Weld DS, Downey D (2022) VILA: Improving Structured Content Extraction from Scientific PDFs Using Visual Layout Groups. Transactions of the Association for Computational Linguistics 10:376–392. https://doi.org/10.1162/tacl_a_00466

Xu Y, Li M, Cui L, Huang S, Wei F, Zhou M (2020) LayoutLM: Pre-training of Text and Layout for Document Image Understanding. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp 1192–1200

> *Written with StackEdit.*