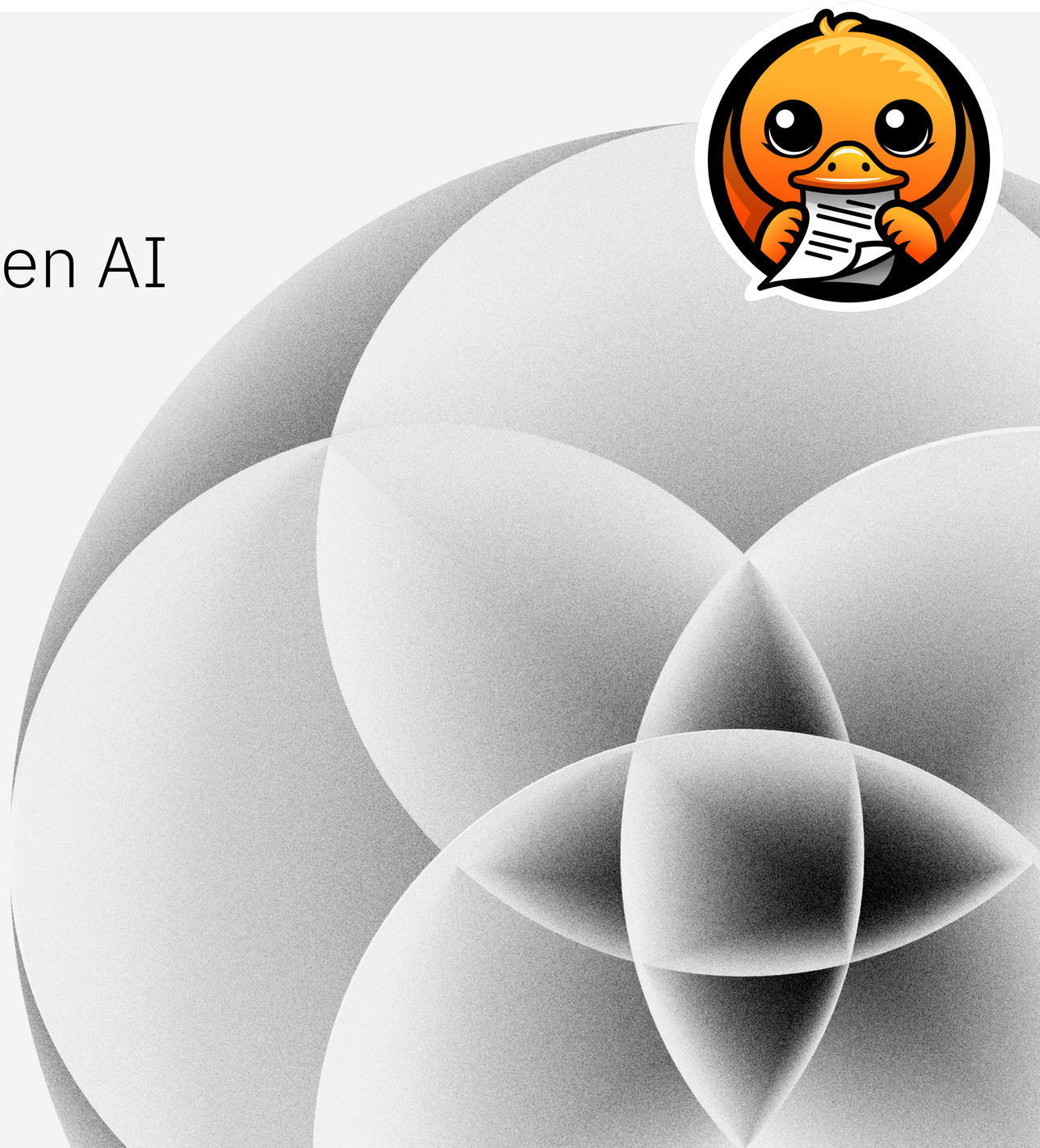
IBM Research

Docling

Get your documents ready for gen AI



–

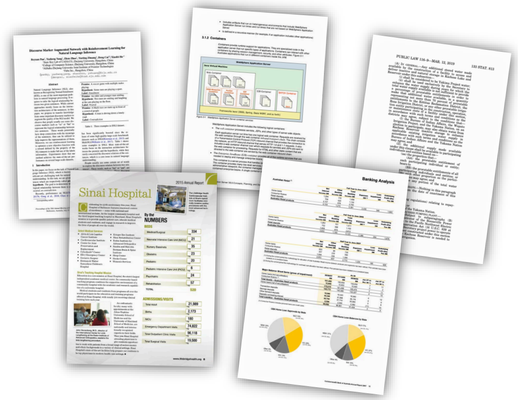
Ming Zhao
Software developer, Open Tech
mingzhao@IBM.com

https://www.linkedin.com/in/mingxuan-z-9a5a6419a/

https://tinyurl.com/pydocling

IBM

Introducing Docling



- 🗂️ Parsing of multiple document formats incl. PDF, DOCX, XLSX, HTML, images, and more
- 📑 Advanced PDF understanding incl. page layout, reading order, table structure, code, formulas, image classification, ...
- 🧬 Unified, expressive DoclingDocument representation format
- ↪ Various export formats (Markdown, HTML, JSON)
- 🔒 Local execution for sensitive data and air-gapped environments
- 🤖 Many plug-and-play ecosystem integrations
- 🔍 Extensive OCR support for scanned PDFs and images
- 🥚 Support of Visual Language Models
- 💻 Simple and convenient CLI

```
pip install docling

# a single document to markdown
docling https://arxiv.org/pdf/2408.09869.pdf

# a folder of documents to markdown and json
docling --to json --to md ./inputs/
```

The story behind Docling

InstructLab integration

Red Hat team uses DS API in the ilab UI

First integrations

Docling is used in data-prep-kit, LlamaIndex, LangChain

New document formats

Docling v2 has native support for docx, pptx, html, and more

InstructLab 0.21

Docling is used for knowledge taxonomy in InstructLab

Early 2024 — June — July 18th — August — Early Oct — Early Nov — Nov 16th — Dec 4th

Deep Search and InstructLab

Deep Search leveraged by InstructLab team for ingesting books

Let's go open-source!

Executive decision to support InstructLab with an open-source version of the DS conversion

Docling is released

Docling goes viral!

+10k GH ⭐ in a week

No. 1 repository in Github.com trending

Most popular IBM open-source

Red Hat Enterprise Linux AI

RHEL AI 1.3

Docling is included in the RHEL AI 1.3 release

IBM

Without Docling...

...it can go bad.

gurovdigital 15 h

lol, over 20 scientific papers now feature the

integrated at pH 7.0. Peptide was released which established that the coats contained substrate for the lytic enzyme present in spores. Peptide was also released from spore coats of *B. megaterium* by the action of the enzyme from *B. cereus* spores. The lytic enzyme did not attack intact resting spores.

The spore develops in the vegetative cell, which thus becomes a sporangium. It is by no means certain what happens to the vegetative cell wall when the spore is released. In *Clostridium* species it appears that at least part of this structure is retained as an outer membrane around the spore. It is the opinion of some workers that the wall of the sporulating cell forms the exosporium which exists as an outer

part of the sporangial wall was dissolved away to allow release of the spore. It appears likely that the exosporium of *B. cereus* does not have a composition similar to that of the vegetative cell wall, from the results obtained by Dr. J. R.

692 12 43

Date syrup (as one of the agricultural wastes) was used to produce bacterial cellulose using Gluconastobacter xylinus. Fourier transform infrared spectroscopy (FTIR), vegetative electron microscopy, and X-ray diffraction were used to determine the structure of bacterial cellulose, cellulose fibers, and crystallinity of the samples (Moosavi an

Silver and gold nanoparticles for [HTML] m

The spore develops in the vegetative cell, which thus becomes a sporangium. It is by no means certain what happens to the vegetative cell wall when the spore is released. In Clostridium species it appears that at least part of this structure is retained as an outer membrane around the spore. It is the opinion of some workers that the wall of the sporulating cell forms the exosporium which exists as an outer coat around spores of several Bacilus species. Spores of several varieties of B. cereus had exosporia whereas these structures appeared to be absent from spores of B. megaterium and B. subtilis. It seems, however, that in Bacillus species at least, the greater part of the vegetative cell wall is dissolved away before the developed spore is released. If this is true, then soluble components containing the characteristic constituents should appear in the medium during spore release. Culture filtrates from B. cereus organisms at various stages of growth and sporulation were hydrolyzed and the hydrolyzates analyzed for amino sugars and diaminopimelic acid (28). Results showed that a large increase in the concentration of these substances in the culture filtrate occurred during spore release (table 2); they were found to be present in a nondialyzable peptide of the characteristic type. It was concluded that at least part of the sporangial wall was dissolved away to allow release of the spore. It appears likely that the exosporium of B. cereus does not have a composition similar to that of the vegetative cell wall, from the results obtained by Dr. J. R. Norris of Leeds University (personal communication). He treated spores with a highly active preparation of lytic enzyme from B. cereus spores and examined the effect by means of electron microscopy. No evidence of lysis of the exosporium was obtained.

[PDF] res

BRONZE PIECES FROM JEYRAN TEPE, UZBAKI

B SODAEI, H RAHNEMA - researchgate.net

This study is a report of the results of metallographic study of 5 bronze pieces found in Jeyrān Tepe dating back to the Iron

Recovering structured content from PDF

with low-level PDF parsers

Table 1: DocLayNet dataset overview. Along with the frequency of each class label, we present the relative occurrence (as % of row "Total") in the train, test and validation sets. The inter-annotator agreement is computed as the mAP@0.5-0.95 metric between pairwise annotations from the triple-annotated pages, from which we obtain accuracy ranges.

		% of Total			triple inter-annotator mAP @ 0.5-0.95 (%)						
class label	Count	Train	Test	Val	All	Fin	Man	Sci	Law	Pat	Ten
Caption	22524	2.04	1.77	2.32	84-89	40-61	86-92	94-99	95-99	69-78	n/a
Footnote	6318	0.60	0.31	0.58	83-91	n/a	100	62-88	85-94	n/a	82-97
Formula	25027	2.25	1.90	2.96	83-85	n/a	n/a	84-87	86-96	n/a	n/a
List-item	185660	17.19	13.34	15.82	87-88	74-83	90-92	97-97	81-85	75-88	93-95
Page-footer	70878	6.51	5.58	6.00	93-94	88-90	95-96	100	92-97	100	96-98
Page-header	58022	5.10	6.70	5.06	85-89	66-76	90-94	98-100	91-92	97-99	81-86
Picture	45976	4.21	2.78	5.31	69-71	56-59	82-86	69-82	80-95	66-71	59-76
Section-header	142884	12.60	15.77	12.85	83-84	76-81	90-92	94-95	87-94	69-73	78-86
Table	34733	3.20	2.27	3.60	77-81	75-80	83-86	98-99	58-80	79-84	70-85
Text	510377	45.82	49.28	45.00	84-86	81-86	88-93	89-93	87-92	71-79	87-95
Title	5071	0.47	0.30	0.50	60-72	24-63	50-63	94-100	82-96	68-79	24-56
Total	1107470	941123	99816	66531	82-83	71-74	79-81	89-94	86-91	71-76	68-85

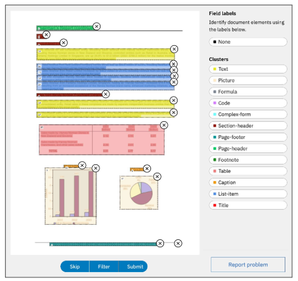


Figure 3: Corpus Conversion Service annotation user interface. The PDF page is shown in the background, with overlaid text-cells (in darker shades). The annotation boxes can be drawn by dragging a rectangle over each segment with the respective label from the palette on the right.

we distributed the annotation workload and performed continuous quality controls. Phase one and two required a small team of experts only. For phases three and four, a group of 40 dedicated annotators were assembled and supervised.

Phase 1: Data selection and preparation. Our inclusion criteria for documents were described in Section 3. A large effort went into ensuring that all documents are free to use. The data sources include publication repositories such as arXiv[3], government offices, company websites as well as data directory services for financial reports and patents. Scanned documents were excluded wherever possible because they can be rotated or skewed. This would not allow us to perform annotation with rectangular bounding-boxes and therefore complicate the annotation process.

Preparation work included uploading and parsing the sourced PDF documents in the Corpus Conversion Service (CCS) [22], a cloud-native platform which provides a visual annotation interface and allows for dataset inspection and analysis. The annotation interface of CCS is shown in Figure 3. The desired balance of pages between the different document categories was achieved by selective subsampling of pages with certain desired properties. For example, we made sure to include the title page of each document and bias the remaining page selection to those with figures or tables. The latter was achieved by leveraging pre-trained object detection models from PubLayNet, which helped us estimate how many figures and tables a given page contains.

Phase 2: Label selection and guideline. We reviewed the collected documents and identified the most common structural features they exhibit. This was achieved by identifying recurrent layout elements and lead us to the definition of 11 distinct class labels. These 11 class labels are *Caption, Footnote, Formula, List-item, Page-footer, Page-header, Picture, Section-header, Table, Text,* and *Title.* Critical factors that were considered for the choice of these class labels were (1) the overall occurrence of the label, (2) the specificity of the label, (3) recognisability on a single page (i.e. no need for context from previous or next page) and (4) overall coverage of the page. Specificity ensures that the choice of label is not ambiguous, while coverage ensures that all meaningful items on a page can be annotated. We refrained from class labels that are very specific to a document category, such as *Abstract* in the *Scientific Articles* category. We also avoided class labels that are tightly linked to the semantics of the text. Labels such as *Author* and *Affiliation*, as seen in DocBank, are often only distinguishable by discriminating on

[3] https://arxiv.org/

! undesired page headers

KDD '22, August 14–18, 2022, Washington, DC, USA Birgit Pfitzmann, Christoph Auer, Michele Dolfi, A
Nassar, and Peter Staar

Table 1: DocLayNet dataset overview. Along with the frequency of each class label, we present the rel
occurrence (as %
of row "Total") in the train, test and validation sets. The inter-annotator agreement is computed as the
0.95 metric
between pairwise annotations from the triple-annotated pages, from which we obtain accuracy range

% of Total

triple inter-annotator mAP @ 0.5-0.95 (%)

[...]
Count
22524
6318
25027
185660
70878
58022
45976
142884
34733
510377
5071
1107470

[...]

include publication repositories such as arXiv3, government o"ces, company websites as well as data directory services for #nancial reports and patents. Scanned documents were excluded wherever possible because they can be rotated or skewed. This would not allow us to perform annotation with rectangular bounding-boxes and therefore complicate the annotation process.

[...]

! Tables not understood

! Image content missing

! Line wraps not understood

! Multi-column often breaks order

✓ Very fast and cheap

✗ Incomplete

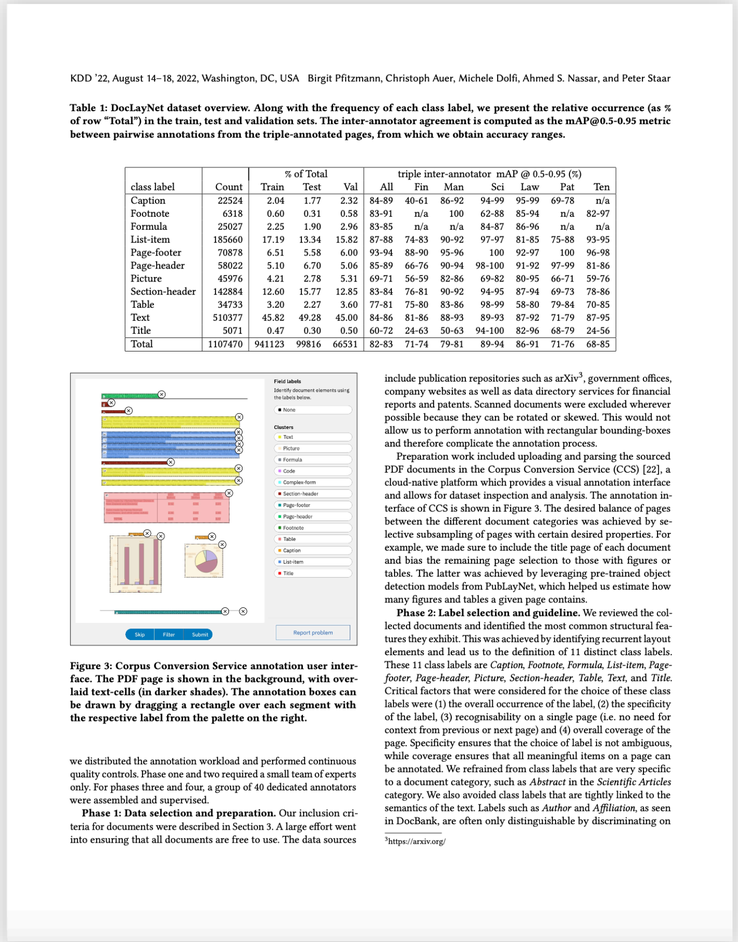
✗ Loss of structure

✗ Noisy

➔ Unfit for most use cases

Recovering structured content from PDF

with Docling



KDD '22, August 14–18, 2022, Washington, DC, USA Birgit Pfitzmann, Christoph Auer, Michele Dolfi, Ahmed S. Nassar, and Peter Staar

Table 1: DocLayNet dataset overview. Along with the frequency of each class label, we present the relative occurrence (as % of row "Total") in the train, test and validation sets. The inter-annotator agreement is computed as the mAP@0.5-0.95 metric between pairwise annotations from the triple-annotated pages, from which we obtain accuracy ranges.

class label	Count	% of Total			triple inter-annotator mAP @ 0.5-0.95 (%)						
		Train	Test	Val	All	Fin	Man	Sci	Law	Pat	Ten
Caption	22524	2.04	1.77	2.32	84-89	40-61	86-92	94-99	95-99	69-78	n/a
Footnote	6318	0.60	0.31	0.58	83-91	n/a	100	62-88	85-94	n/a	82-97
Formula	25027	2.25	1.90	2.96	83-85	n/a	n/a	84-87	86-96	n/a	n/a
List-item	185660	17.19	13.34	15.82	87-88	74-83	90-92	97-97	81-85	75-88	93-95
Page-footer	70878	6.51	5.58	6.00	93-94	88-90	95-96	100	92-97	100	96-98
Page-header	58022	5.10	6.70	5.06	85-89	66-76	90-94	98-100	91-92	97-99	81-86
Picture	45976	4.21	2.78	5.31	69-71	56-59	82-86	69-82	80-95	66-71	59-76
Section-header	142884	12.60	15.77	12.85	83-84	76-81	90-92	94-95	87-94	69-73	78-86
Table	34733	3.20	2.27	3.60	77-81	75-80	83-86	98-99	58-80	79-84	70-85
Text	510377	45.82	49.28	45.00	84-86	81-86	88-93	89-93	87-92	71-79	87-95
Title	5071	0.47	0.30	0.50	60-72	24-63	50-63	94-100	82-96	68-79	24-56
Total	1107470	941123	99816	66531	82-83	71-74	79-81	89-94	86-91	71-76	68-85

Figure 3: Corpus Conversion Service annotation user interface. The PDF page is shown in the background, with overlaid text-cells (in darker shades). The annotation boxes can be drawn by dragging a rectangle over each segment with the respective label from the palette on the right.

we distributed the annotation workload and performed continuous quality controls. Phase one and two required a small team of experts only. For phases three and four, a group of 40 dedicated annotators were assembled and supervised.

Phase 1: Data selection and preparation. Our inclusion criteria for documents were described in Section 3. A large effort went into ensuring that all documents are free to use. The data sources include publication repositories such as arXiv[3], government offices, company websites as well as data directory services for financial reports and patents. Scanned documents were excluded wherever possible because they can be rotated or skewed. This would not allow us to perform annotation with rectangular bounding-boxes and therefore complicate the annotation process.

Preparation work included uploading and parsing the sourced PDF documents in the Corpus Conversion Service (CCS) [22], a cloud-native platform which provides a visual annotation interface and allows for dataset inspection and analysis. The annotation interface of CCS is shown in Figure 3. The desired balance of pages between the different document categories was achieved by selective subsampling of pages with certain desired properties. For example, we made sure to include the title page of each document and bias the remaining page selection to those with figures or tables. The latter was achieved by leveraging pre-trained object detection models from PubLayNet, which helped us estimate how many figures and tables a given page contains.

Phase 2: Label selection and guideline. We reviewed the collected documents and identified the most common structural features they exhibit. This was achieved by identifying recurrent layout elements and lead us to the definition of 11 distinct class labels. These 11 class labels are *Caption, Footnote, Formula, List-item, Page-footer, Page-header, Picture, Section-header, Table, Text,* and *Title.* Critical factors that were considered for the choice of these class labels were (1) the overall occurrence of the label, (2) the specificity of the label, (3) recognisability on a single page (i.e. no need for context from previous or next page) and (4) overall coverage of the page. Specificity ensures that the choice of label is not ambiguous, while coverage ensures that all meaningful items on a page can be annotated. We refrained from class labels that are very specific to a document category, such as *Abstract* in the *Scientific Articles* category. We also avoided class labels that are tightly linked to the semantics of the text. Labels such as *Author* and *Affiliation*, as seen in DocBank, are often only distinguishable by discriminating on

[3]https://arxiv.org/

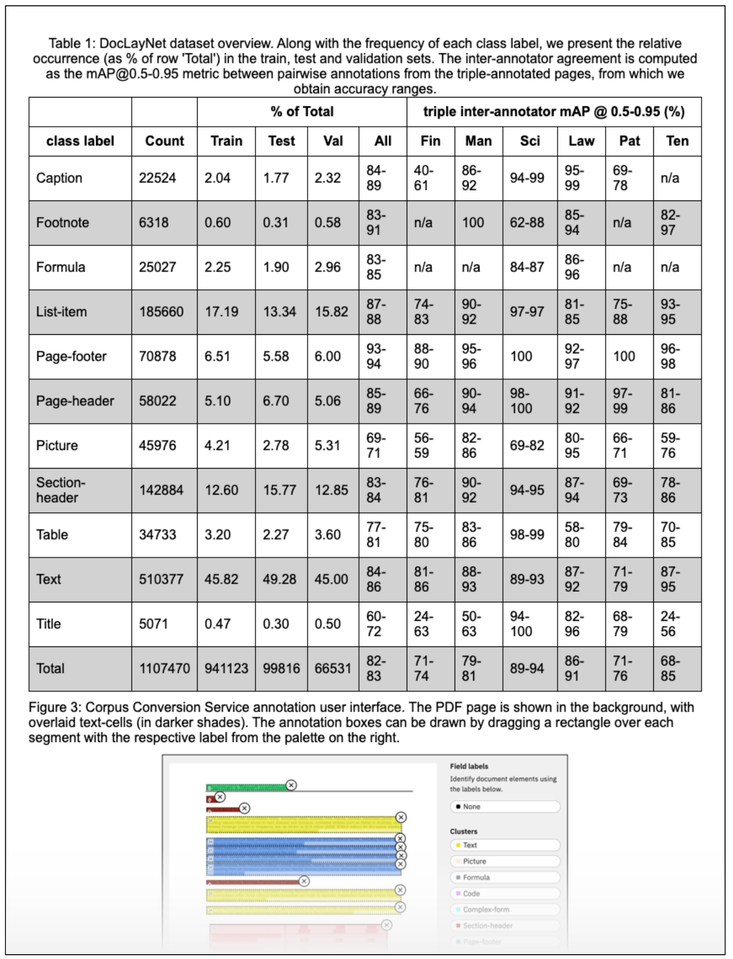


Table 1: DocLayNet dataset overview. Along with the frequency of each class label, we present the relative occurrence (as % of row 'Total') in the train, test and validation sets. The inter-annotator agreement is computed as the mAP@0.5-0.95 metric between pairwise annotations from the triple-annotated pages, from which we obtain accuracy ranges.

		% of Total				triple inter-annotator mAP @ 0.5-0.95 (%)					
class label	Count	Train	Test	Val	All	Fin	Man	Sci	Law	Pat	Ten
Caption	22524	2.04	1.77	2.32	84-89	40-61	86-92	94-99	95-99	69-78	n/a
Footnote	6318	0.60	0.31	0.58	83-91	n/a	100	62-88	85-94	n/a	82-97
Formula	25027	2.25	1.90	2.96	83-85	n/a	n/a	84-87	86-96	n/a	n/a
List-item	185660	17.19	13.34	15.82	87-88	74-83	90-92	97-97	81-85	75-88	93-95
Page-footer	70878	6.51	5.58	6.00	93-94	88-90	95-96	100	92-97	100	96-98
Page-header	58022	5.10	6.70	5.06	85-89	66-76	90-94	98-100	91-92	97-99	81-86
Picture	45976	4.21	2.78	5.31	69-71	56-59	82-86	69-82	80-95	66-71	59-76
Section-header	142884	12.60	15.77	12.85	83-84	76-81	90-92	94-95	87-94	69-73	78-86
Table	34733	3.20	2.27	3.60	77-81	75-80	83-86	98-99	58-80	79-84	70-85
Text	510377	45.82	49.28	45.00	84-86	81-86	88-93	89-93	87-92	71-79	87-95
Title	5071	0.47	0.30	0.50	60-72	24-63	50-63	94-100	82-96	68-79	24-56
Total	1107470	941123	99816	66531	82-83	71-74	79-81	89-94	86-91	71-76	68-85

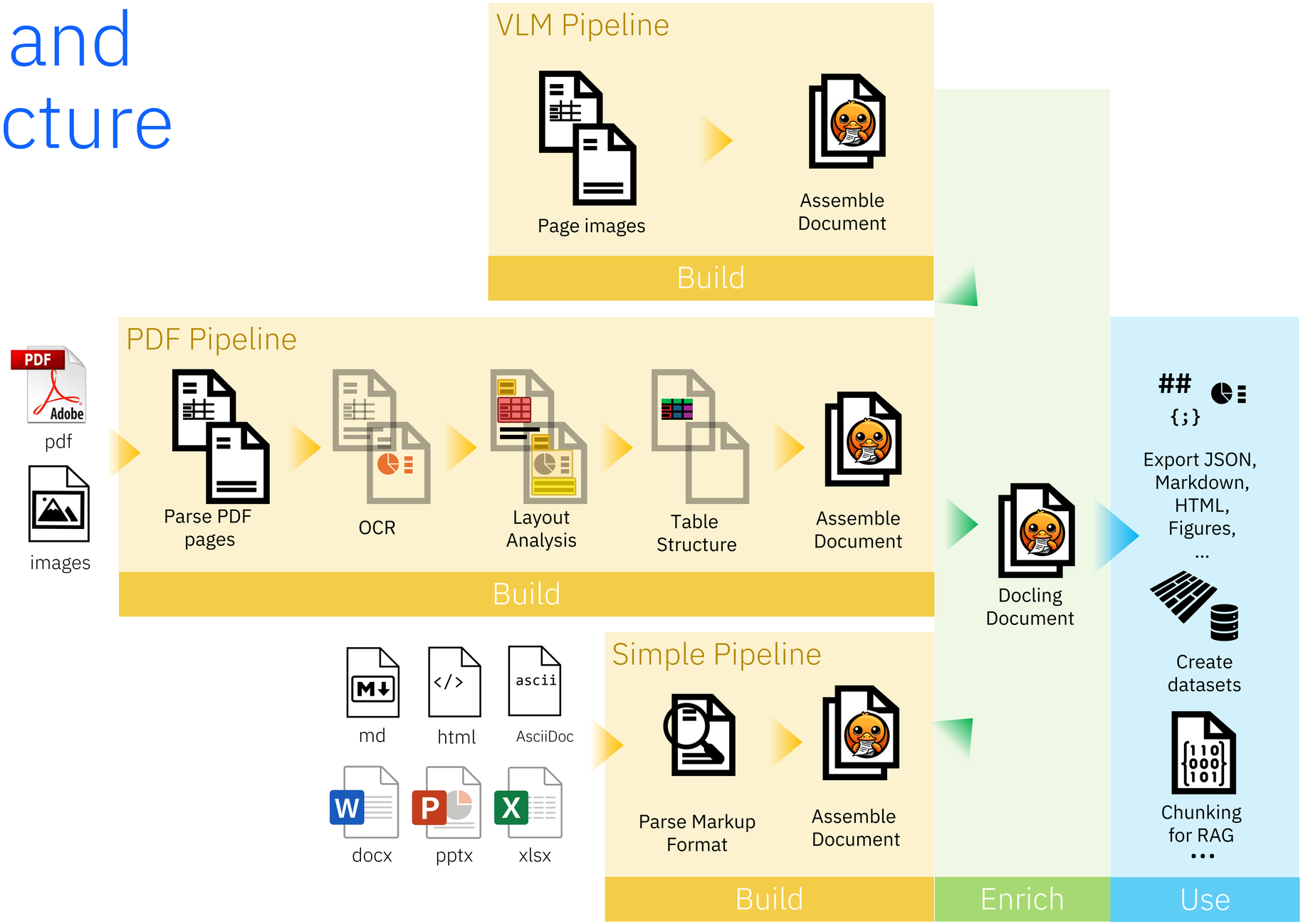
Figure 3: Corpus Conversion Service annotation user interface. The PDF page is shown in the background, with overlaid text-cells (in darker shades). The annotation boxes can be drawn by dragging a rectangle over each segment with the respective label from the palette on the right.

*results rendered as HTML for visualization purposes

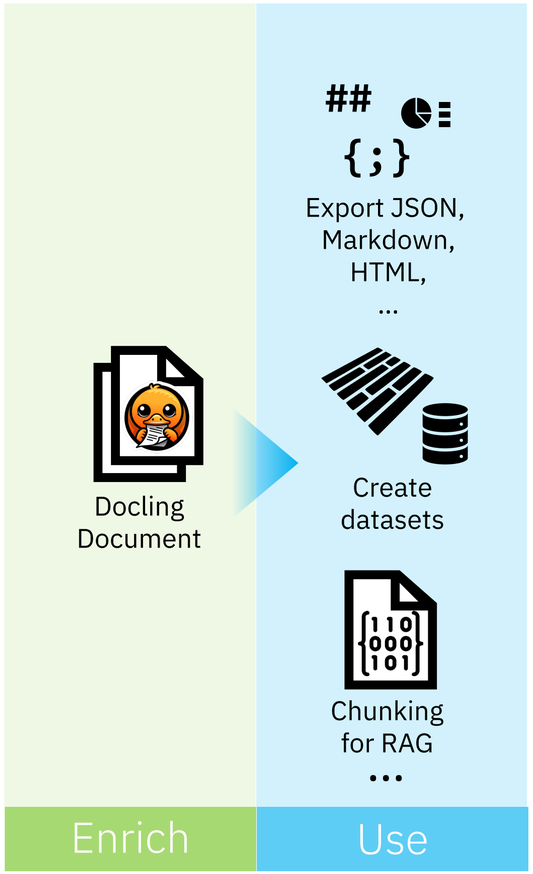
- ✓ Good quality
- ✓ Fast and cheap
- ✓ Fully local operation
- ✓ Structured format output

- → Cost-effective at scale, with consistent representation and high quality

Design and Architecture



DoclingDocument data model




```yaml
version: 1.0.0
schema_name: DoclingDocument

body: # The root node of the document content (excluding headers, footers, ...)
  children:
  - $ref: '#/texts/0' # text: Summer activities
  - $ref: '#/texts/1' # title: Swimming in the lake
  label: unspecified
  name: _root_
  self_ref: '#/body'

texts: # The plain text items in this document.
- self_ref: '#/texts/0'
  orig: Summer activities
  text: Summer activities
  label: paragraph # The semantics of a text element are represented by the label
  children: []
  parent:
    $ref: '#/body'
  prov: []
- self_ref: '#/texts/1'
  orig: Swimming in the lake
  text: Swimming in the lake
  label: title
  children: # Any item can have children to reflect section hierachy
  - $ref: '#/texts/2' # text: Duck
  - $ref: '#/texts/3' # text: (empty text)
  - $ref: '#/texts/4' # text: Figure 1: This is a cute duckling
  - $ref: '#/texts/5' # section_header: Let's swim!
  # ...
  parent:
    $ref: '#/body'
  prov: []
# ...
```

Summer activities

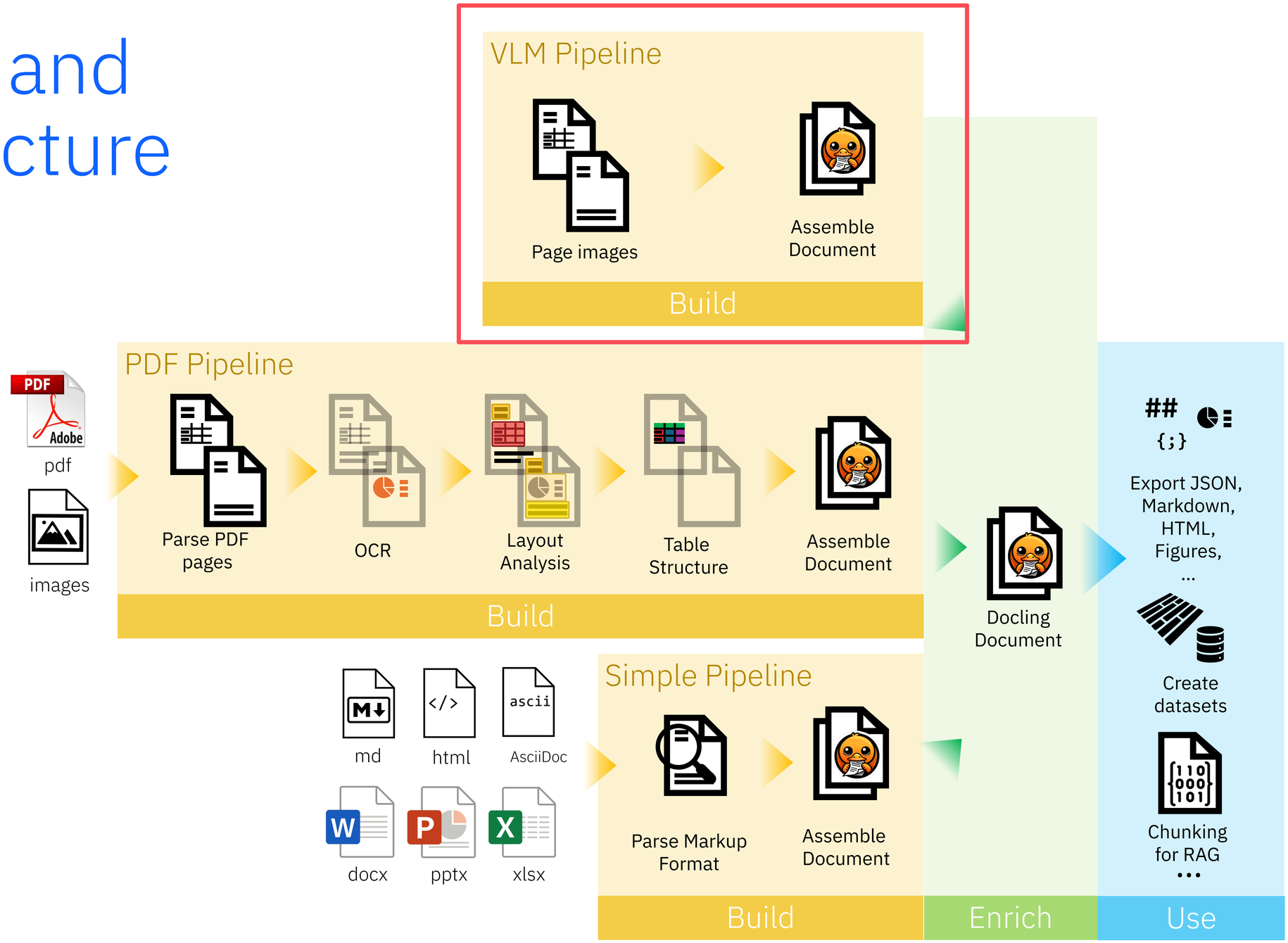
Swimming in the lake

Duck

Figure 1: This is a cute duckling

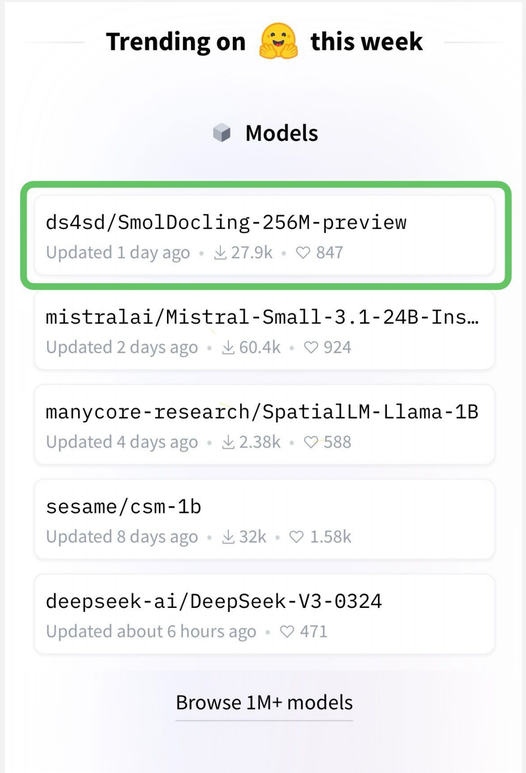
Let’s swim!

Design and Architecture



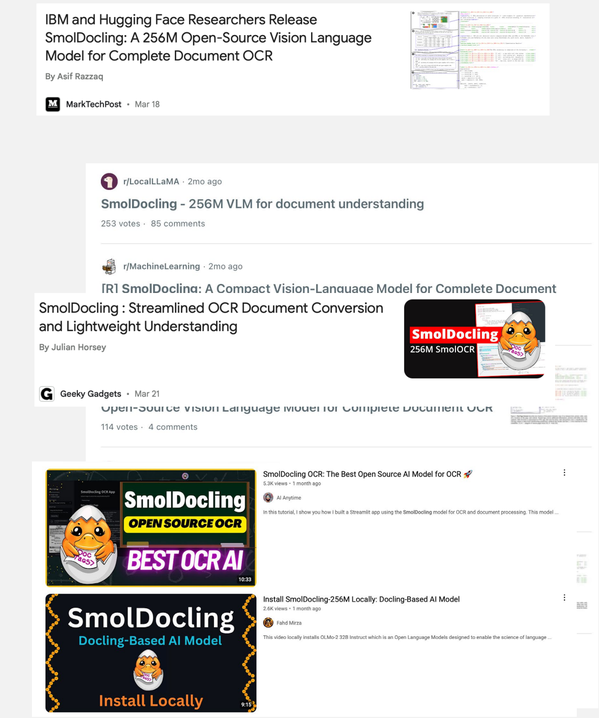
SmolDocling

SmolDocling made a strong debut, generating excitement across the document understanding community.



Total downloads

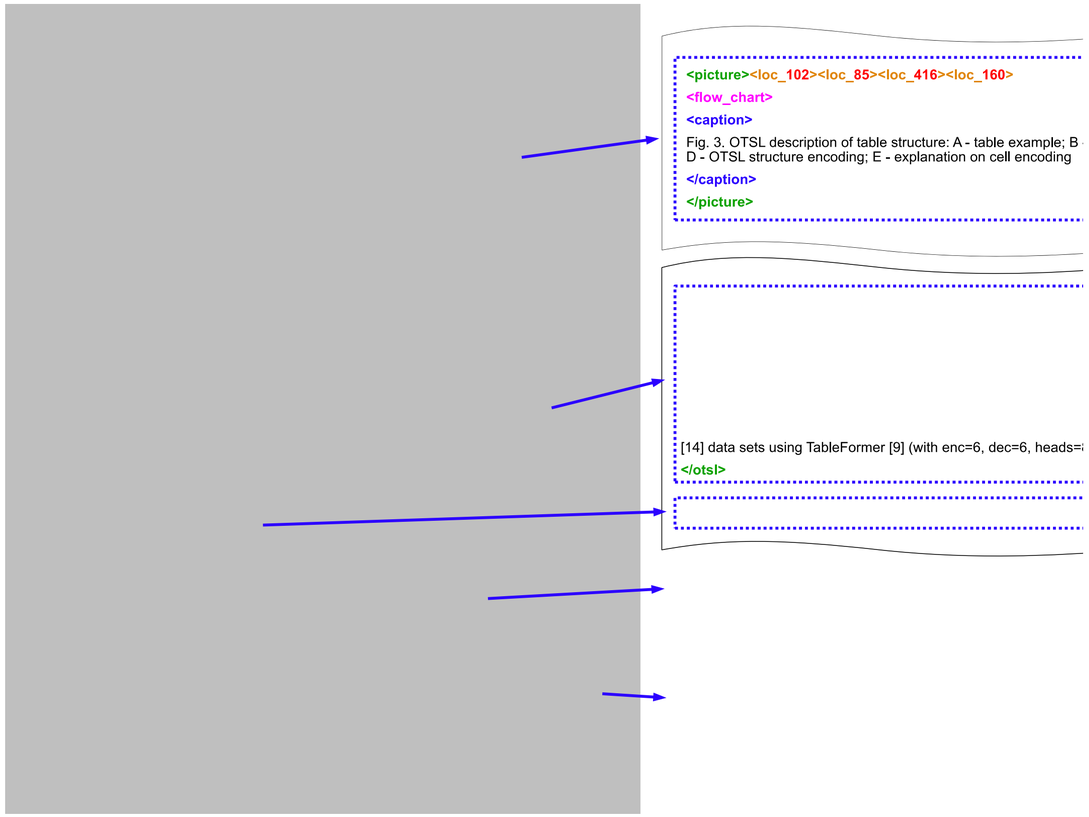
+155k



https://huggingface.co/ds4sd/SmolDocling-256M-preview

SmolDocling

Why it works – and why people love it.

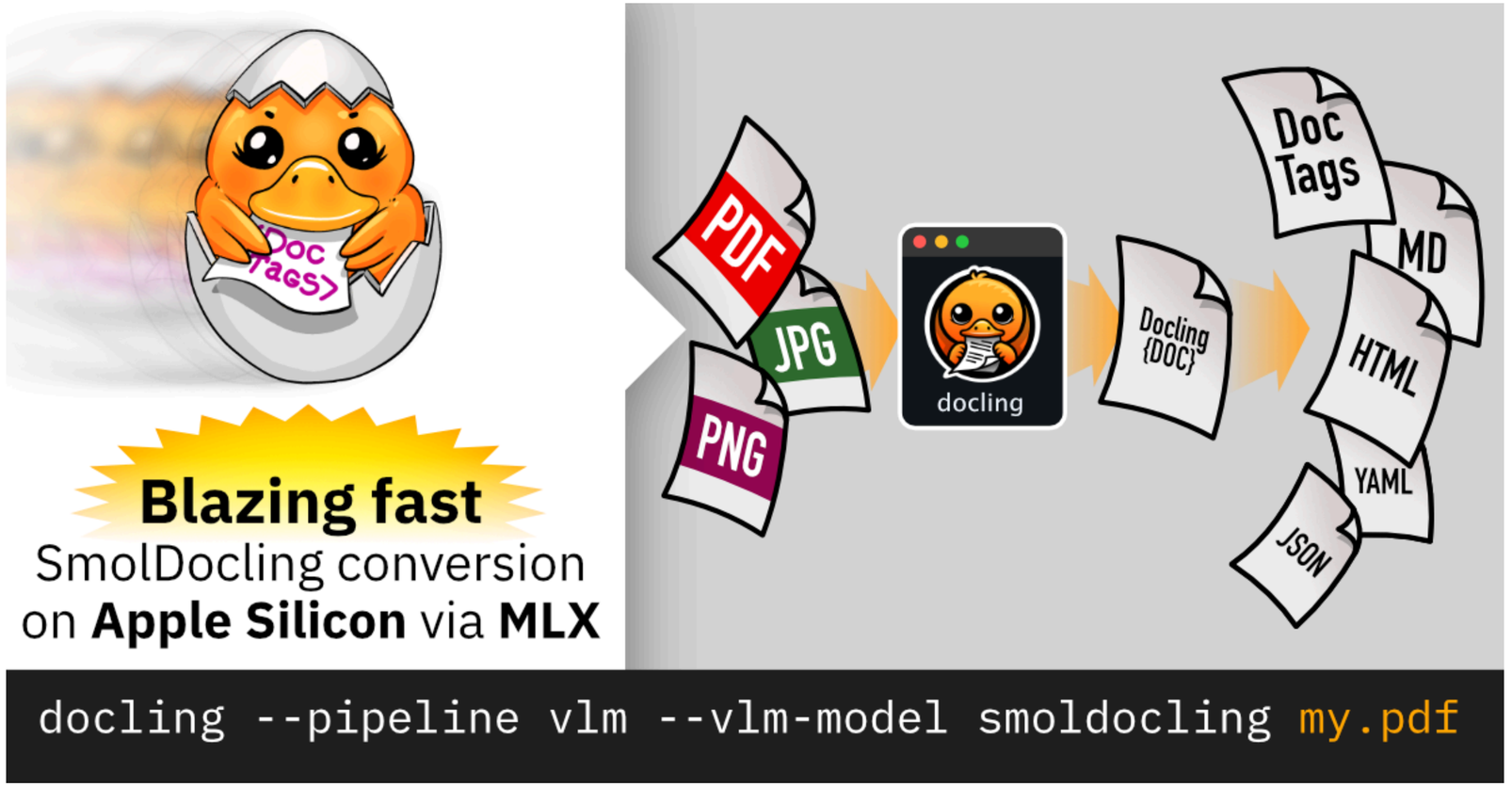


- 🔍 **OCR & Bounding Boxes** – Accurate text extraction with region-level precision.
- 🏷️ **DocTags** – Minimal, efficient, and fully compatible with DoclingDocuments.
- 📐 **Layout Awareness** – Preserves structure, positions, and element bounding boxes.
- 💻 **Code, Formula, Figures** – Recognizes complex structured content.
- 📊 **Charts & Tables** – Parses and reconstructs structured data visually.
- 📝 **Captions, Lists, & Headers**– Maintains hierarchy and semantic links.
- 📄 **Full-Page Conversion** – Converts complete pages with all content types.
- 📂 **General Document Support** – Trained on diverse document types.
- ⚡ **Fast Inference** – ~0.35 sec/page on A100 with VLLM.

https://huggingface.co/ds4sd/SmolDocling-256M-preview

SmolDocling

ultra-compact vision-language model for end-to-end document conversion



https://huggingface.co/ds4sd/SmolDocling-256M-preview

Source

order to compute the TED score. Inference timing results for all experiments were obtained from the same machine on a single core with AMD EPYC 7763 CPU @2.45 GHz.

5.1 Hyper Parameter Optimization

We have chosen the PubTabNet data set to perform HPO, since it includes a highly diverse set of tables. Also we report TED scores separately for simple and complex tables (tables with cell spans). Results are presented in Table. 1. It is evident that with OTSL, our model achieves the same TED score and slightly better mAP scores in comparison to HTML. However OTSL yields a *2x speed up* in the inference runtime over HTML.

Table 1. HPO performed in OTSL and HTML representation on the same transformer-based TableFormer [9] architecture, trained only on PubTabNet [22]. Effects of reducing the # of layers in encoder and decoder stages of the model show that smaller models trained on OTSL perform better, especially in recognizing complex table structures, and maintain a much higher mAP score than the HTML counterpart.

# enc-layers	# dec-layers	Language	TEDs			mAP (0.75)	Inference time (secs)
			simple	complex	all		
6	6	OTSL	0.965	0.934	0.955	**0.88**	**2.73**
		HTML	0.969	0.927	0.955	0.857	5.39
4	4	OTSL	0.938	0.904	0.927	**0.853**	**1.97**
		HTML	0.952	0.909	**0.938**	0.843	3.77
2	4	OTSL	0.923	0.897	0.915	**0.859**	**1.91**
		HTML	0.945	0.901	**0.931**	0.834	3.81
4	2	OTSL	0.952	0.92	**0.942**	**0.857**	**1.22**
		HTML	0.944	0.903	0.931	0.824	2

5.2 Quantitative Results

We picked the model parameter configuration that produced the best prediction quality (enc=6, dec=6, heads=8) with PubTabNet alone, then independently trained and evaluated it on three publicly available data sets: PubTabNet (395k samples), FinTabNet (113k samples) and PubTables-1M (about 1M samples). Performance results are presented in Table. 2. It is clearly evident that the model trained on OTSL outperforms HTML across the board, keeping high TEDs and mAP scores even on difficult financial tables (FinTabNet) that contain sparse and large tables.

Additionally, the results show that OTSL has an advantage over HTML when applied on a bigger data set like PubTables-1M and achieves significantly improved scores. Finally, OTSL achieves faster inference due to fewer decoding steps which is a result of the reduced sequence representation.

Qwen2.5-3B (25 sec)

Optimized Table Tokenization for Table Structure Recognition

5.1 Hyper Parameter Parameter Optimization

We have chosen the PubTabNet data set to perform HPO, since it includes a highly diverse set of tables. Also, we report TED scores separately for simple and complex tables (tables with cell spans). Results are presented in Table 1. It is evident that with OTSL, our model achieves the same TED score and slightly better mAP scores in comparison to HTML. However, OTSL yields a 2x speed up in the inference runtime over HTML.

Table 1. HPO performed in OTSL and HTML representation on the same representation architecture, trained only on PubTabNet. Effects of reducing the # of layers in encoder and decoder stages of the model show that smaller models trained on OTSL perform better, especially in recognizing complex table structures, and maintain a much higher mAP score than the HTML counterpart.

# enc-layers	# dec-layers	Language	TEDs	mAP	Inference time (sec)
6	6	OTSL	0.965	0.934	0.955
6	6	HTML	0.969	0.927	0.955
4	4	OTSL	0.938	0.904	0.927
4	4	HTML	0.952	0.909	0.938
2	2	OTSL	0.923	0.897	0.915
2	2	HTML	0.945	0.901	0.931
4	2	OTSL	0.952	0.92	0.942
4	2	HTML	0.944	0.903	0.931

5.2 Quantitative Results

We picked the model parameter configuration that produced the best prediction quality (enc=6, dec=6, heads=8) with PubTabNet alone, then independently trained and evaluated it on three publicly available data sets: PubTabNet (395k samples), FinTabNet (113k samples) and PubTables-1M (about 1M samples). Performance results are presented in Table 2. It is clearly evident that the model trained on OTSL outperforms HTML across the board, keeping high TEDs and mAP scores even on difficult financial tables (FinTabNet) that contain sparse and large tables.

Additionally, the results show that OTSL has an advantage over HTML when applied on a bigger data set like PubTables-1M and achieves significantly improved scores. Finally, OTSL achieves faster inference due to fewer decoding steps, which is a result of the reduced sequence representation.

Phi4-14B (350 sec)

Certainly! Below is the markdown table converted from the image provided:

```
| # enc-layers | # dec-layers | Language | simple | complex | all | mAP | Inference time (secs) |
|--------------|--------------|----------|--------|---------|-----|-----|-----------------------|
| 6            | 6            | OTSL     | 0.965  | 0.934   | 0.955 | 0.88 | 2.73                 |
|              |              | HTML     | 0.969  | 0.927   | 0.955 | 0.857 | 5.39                |
| 4            | 4            | OTSL     | 0.938  | 0.904   | 0.927 | 0.853 | 1.97                |
|              |              | HTML     | 0.952  | 0.909   | 0.938 | 0.843 | 3.77                |
| 2            | 4            | OTSL     | 0.923  | 0.897   | 0.915 | 0.859 | 1.91                |
|              |              | HTML     | 0.945  | 0.901   | 0.931 | 0.834 | 3.81                |
| 4            | 2            | OTSL     | 0.952  | 0.92    | 0.942 | 0.857 | 1.22                |
|              |              | HTML     | 0.944  | 0.903   | 0.931 | 0.824 | 2                   |
```

Please note that the table is titled "Table 1. HPO performed in OTSL and HTML representation on the same transform based TableFormer architecture, trained only on PubTabNet [22]. Effects of reducing the # of layers in encoder and decoder stages of the model show that smaller models trained on OTSL perform better, especially in recognizing complex table structures, and maintain a much higher mAP score than the HTML counterpart."

Pixtral 12B (287 sec)

Optimized Table Tokenization for Table Structure Recognition

order to compute the TED score. Inference timing results for all experiments were obtained from the same machine on a single core with AMD EPYC 7763 CPU @2.45 GHz.

5.1 Hyper Parameter Optimization

We have chosen the PubTabNet data set to perform HPO, since it includes a highly diverse set of tables. Also we report TED scores separately for simple and complex tables (tables with cell spans). Results are presented in Table. 1. It is evident that with OTSL, our model achieves the same TED score and slightly better mAP scores in comparison to HTML. However OTSL yields a 2x speed up in the inference runtime over HTML.

#	#	Language	TEDs		mAP	Inference
enc-layers	dec-layers		simple	complex		all
6	6	OTSL	0.965	0.934	0.955	
HTML	0.969	0.927	0.955	0.857	5.39	
4	4	OTSL	0.938	0.904	0.927	
HTML	0.952	0.909	0.938	0.843	3.77	
2	4	OTSL	0.923	0.897	0.915	
HTML	0.945	0.901	0.931	0.834	3.81	
4	2	OTSL	0.952	0.92	0.942	
HTML	0.944	0.903	0.931	0.824	2	

Table 1. HPO performed in OTSL and HTML representation on the same transformer-based TableFormer [8] architecture, trained only on PubTabNet [22]. Effects of reducing the # of layers in encoder and decoder stages of the model show that smaller models trained on OTSL perform better, especially in recognizing complex table structures, and maintain a much higher mAP score than the HTML counterpart.

5.2 Quantitative Results

We picked the model parameter configuration that produced the best prediction quality (enc=6, dec=6, heads=8) with PubTabNet alone, then independently trained and evaluated it on three publicly available data sets: PubTabNet (395k samples), FinTabNet (113k samples) and PubTables-1M (about 1M samples). Performance results are presented in Table 2. It is clearly evident that the model trained on OTSL outperforms HTML across the board, keeping high TEDs and mAP scores even on difficult financial tables (FinTabNet) that contain sparse and large tables.

Additionally, the results show that OTSL has an advantage over HTML when applied on a bigger data set like PubTables-1M and achieves significantly improved scores. Finally, OTSL achieves faster inference due to fewer decoding steps which is a result of the reduced sequence representation.

Source

order to compute the TED score. Inference timing results for all experiments were obtained from the same machine on a single core with AMD EPYC 7763 CPU @2.45 GHz.

5.1 Hyper Parameter Optimization

We have chosen the PubTabNet data set to perform HPO, since it includes a highly diverse set of tables. Also we report TED scores separately for simple and complex tables (tables with cell spans). Results are presented in Table. 1. It is evident that with OTSL, our model achieves the same TED score and slightly better mAP scores in comparison to HTML. However OTSL yields a 2x *speed up* in the inference runtime over HTML.

Table 1. HPO performed in OTSL and HTML representation on the same transformer-based TableFormer [9] architecture, trained only on PubTabNet [22]. Effects of reducing the # of layers in encoder and decoder stages of the model show that smaller models trained on OTSL perform better, especially in recognizing complex table structures, and maintain a much higher mAP score than the HTML counterpart.

# enc-layers	# dec-layers	Language	TEDs simple	TEDs complex	TEDs all	mAP (0.75)	Inference time (secs)
6	6	OTSL	0.965	0.934	0.955	**0.88**	**2.73**
		HTML	0.969	0.927	0.955	0.857	5.39
4	4	OTSL	0.938	0.904	0.927	**0.853**	**1.97**
		HTML	0.952	0.909	**0.938**	0.843	3.77
2	4	OTSL	0.923	0.897	0.915	**0.859**	**1.91**
		HTML	0.945	0.901	**0.931**	0.834	3.81
4	2	OTSL	0.952	0.92	**0.942**	**0.857**	**1.22**
		HTML	0.944	0.903	0.931	0.824	2

5.2 Quantitative Results

We picked the model parameter configuration that produced the best prediction quality (enc=6, dec=6, heads=8) with PubTabNet alone, then independently trained and evaluated it on three publicly available data sets: PubTabNet (395k samples), FinTabNet (113k samples) and PubTables-1M (about 1M samples). Performance results are presented in Table. 2. It is clearly evident that the model trained on OTSL outperforms HTML across the board, keeping high TEDs and mAP scores even on difficult financial tables (FinTabNet) that contain sparse and large tables.

Additionally, the results show that OTSL has an advantage over HTML when applied on a bigger data set like PubTables-1M and achieves significantly improved scores. Finally, OTSL achieves faster inference due to fewer decoding steps which is a result of the reduced sequence representation.

SmolDocling (4 sec)

order to compute the TED score. Inference timing results for all experiments were obtained from the same machine on a single core with AMD EPYC 7763 CPU @2.45 GHz.

5.1 Hyper Parameter Optimization

We have chosen the PubTabNet data set to perform HPO, since it includes a highly diverse set of tables. Also we report TED scores separately for simple and complex tables (tables with cell spans). Results are presented in Table. 1. It is evident that with OTSL, our model achieves the same TED score and slightly better mAP scores in comparison to HTML. However OTSL yields a 2x speed up in the inference runtime over HTML.

Table 1. HPO performed in OTSL and HTML representation on the same transformer-based TableFormer [9] architecture, trained only on PubTabNet [22]. Effectors of reducing the # of layers in encoder and decoder stages of the model show that smaller models trained on OTSL perform better, especially in recognizing complex table structures, and maintain a much higher mAP score than the HTML counterpart.

#	#	Language	TEDs			mAP	Inference (0.75)
enc-layers	dec-layers		simple	complex	all	(0.75)	time (secs)
6	6	OTSL	0.965	0.934	0.955	0.88	2.73
		HTML	0.969	0.927	0.955	0.857	5.39
4	4	OTSL	0.938	0.904	0.927	0.853	1.97
		HTML	0.952	0.909	0.938	0.843	3.77
2	4	OTSL	0.923	0.897	0.915	0.859	1.91
		HTML	0.945	0.901	0.931	0.834	3.81
4	2	OTSL	0.952	0.92	0.942	0.857	1.22
		HTML	0.944	0.903	0.931	0.824	2

5.2 Quantitative Results

We picked the model parameter configuration that produced the best prediction quality (enc=6, dec=6, heads=8) with PubTabNet alone, then independently trained and evaluated it on three publicly available data sets: PubTabNet (395k samples), FinTabNet (113k samples) and PubTables-1M (about 1M samples). Performance results are presented in Table. 2. It is clearly evident that the model trained on OTSL outperforms HTML across the board, keeping high TEDs and mAP scores even on difficult financial tables (FinTabNet) that contain sparse and large tables.

Additionally, the results show that OTSL has an advantage over HTML when applied on a bigger data set like PubTables-1M and achieves significantly improved scores. Finally, OTSL achieves faster inference due to fewer decoding steps which is a result of the reduced sequence representation.

SmolDocling

Stay tuned!

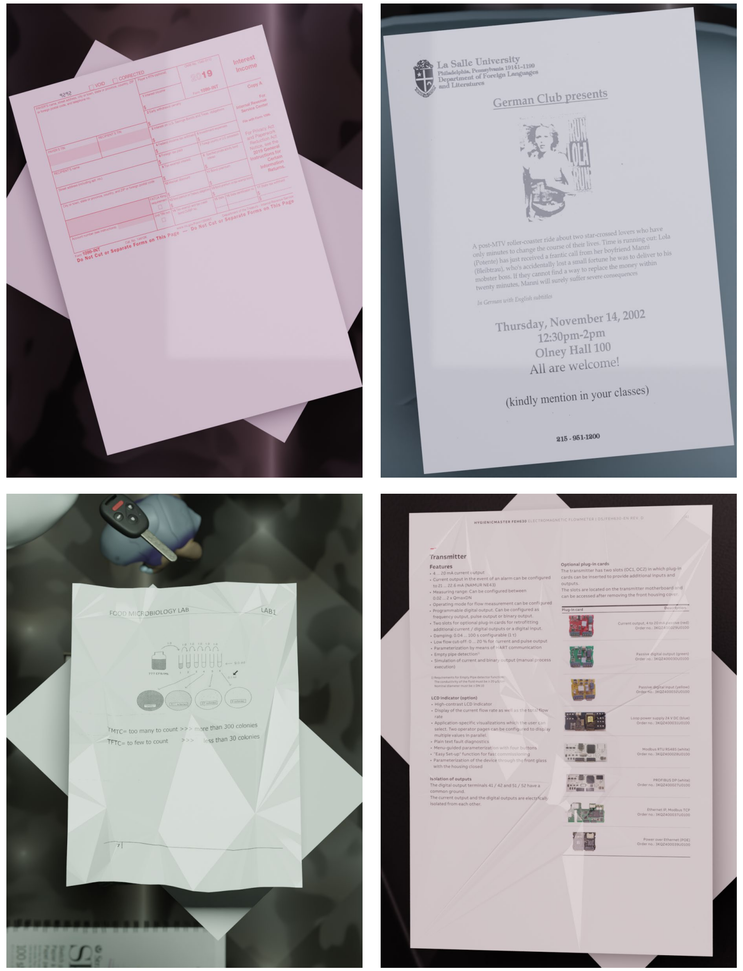
🌐 Multi-lingual support: German, Arabic, French, Spanish, Polish, Chinese and Japanese.

📄 DoclingDocument to PDF pipeline.

🧠 Synthetically generate reasoning and CoT datasets for document understanding.

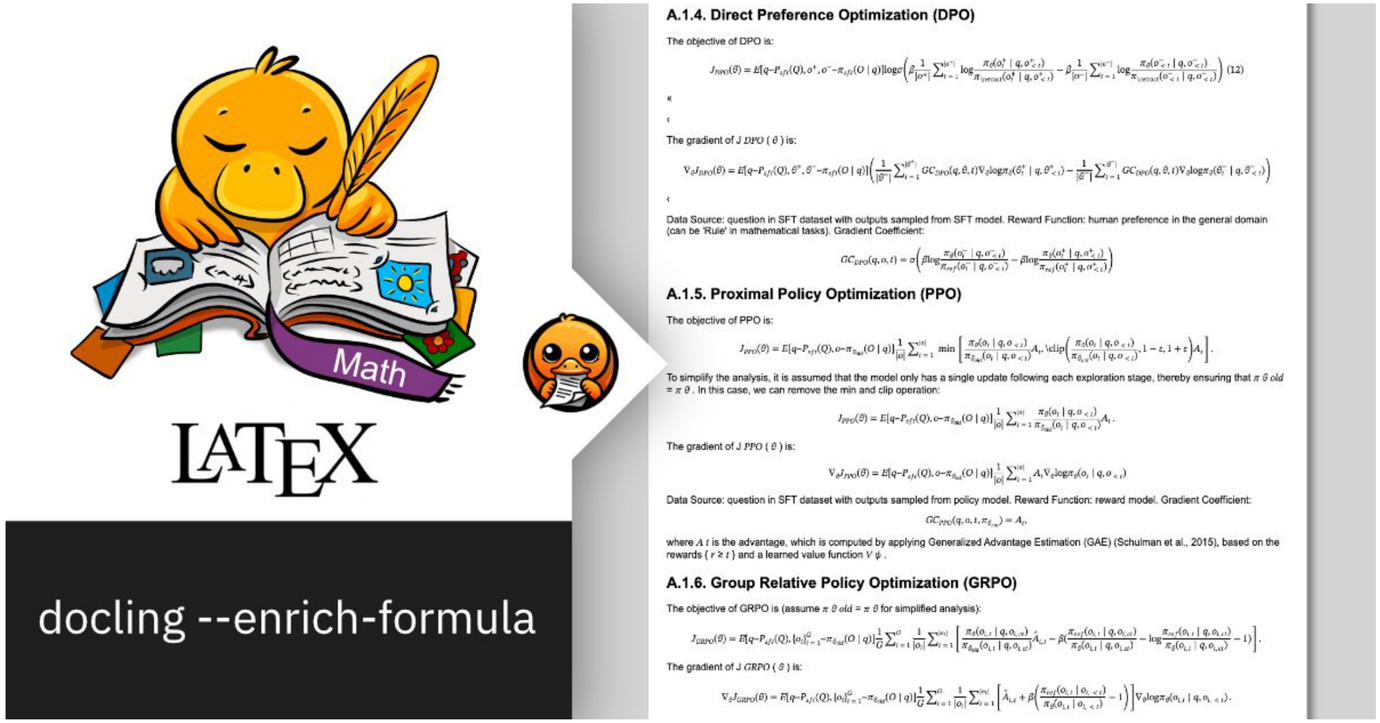
🕵️‍♂️ Text based instruction dataset for understanding DoclingDocuments.

🤖 Synthetically generate documents in the wild.



Enrichment models

Code and Formula model



A.1.4. Direct Preference Optimization (DPO)

The objective of DPO is:

$$J_{DPO}(\theta) = E[q\sim P_{sft}(Q), o^+, o^-\sim\pi_{sft}(O\mid q)]\log\sigma\left(\beta\frac{1}{|o^+|}\sum_{t=1}^{|o^+|}\log\frac{\pi_\theta(o^+_t\mid q,o^+_{<t})}{\pi_{ref(init)}(o^+_t\mid q,o^+_{<t})}-\beta\frac{1}{|o^-|}\sum_{t=1}^{|o^-|}\log\frac{\pi_\theta(o^-_{<t+1}\mid q,o^-_{<t})}{\pi_{ref(init)}(o^-_{<t+1}\mid q,o^-_{<t})}\right) \quad (12)$$

«

‹

The gradient of J *DPO* (θ) is:

$$\nabla_\theta J_{DPO}(\theta) = E[q\sim P_{sft}(Q), o^+, o^-\sim\pi_{sft}(O\mid q)]\left(\frac{1}{|o^+|}\sum_{t=1}^{|o^+|}GC_{DPO}(q,o,t)\nabla_\theta\log\pi_\theta(o^+_t\mid q,o^+_{<t})-\frac{1}{|o^-|}\sum_{t=1}^{|o^-|}GC_{DPO}(q,o,t)\nabla_\theta\log\pi_\theta(o^-_t\mid q,o^-_{<t})\right)$$

‹

Data Source: question in SFT dataset with outputs sampled from SFT model. Reward Function: human preference in the general domain (can be 'Rule' in mathematical tasks). Gradient Coefficient:

$$GC_{DPO}(q,o,t) = \sigma\left(\beta\log\frac{\pi_\theta(o^-_t\mid q,o^-_{<t})}{\pi_{ref}(o^-_t\mid q,o^-_{<t})}-\beta\log\frac{\pi_\theta(o^+_t\mid q,o^+_{<t})}{\pi_{ref}(o^+_t\mid q,o^+_{<t})}\right)$$

A.1.5. Proximal Policy Optimization (PPO)

The objective of PPO is:

$$J_{PPO}(\theta) = E[q\sim P_{sft}(Q), o\sim\pi_{\theta_{old}}(O\mid q)]\frac{1}{|o|}\sum_{t=1}^{|o|}\min\left[\frac{\pi_\theta(o_t\mid q,o_{<t})}{\pi_{\theta_{old}}(o_t\mid q,o_{<t})}A_t,\text{clip}\left(\frac{\pi_\theta(o_t\mid q,o_{<t})}{\pi_{\theta_{old}}(o_t\mid q,o_{<t})},1-\varepsilon,1+\varepsilon\right)A_t\right].$$

To simplify the analysis, it is assumed that the model only has a single update following each exploration stage, thereby ensuring that $\pi_{\theta_{old}} = \pi_\theta$. In this case, we can remove the min and clip operation:

$$J_{PPO}(\theta) = E[q\sim P_{sft}(Q), o\sim\pi_{\theta_{old}}(O\mid q)]\frac{1}{|o|}\sum_{t=1}^{|o|}\frac{\pi_\theta(o_t\mid q,o_{<t})}{\pi_{\theta_{old}}(o_t\mid q,o_{<t})}A_t.$$

The gradient of J *PPO* (θ) is:

$$\nabla_\theta J_{PPO}(\theta) = E[q\sim P_{sft}(Q), o\sim\pi_{\theta_{old}}(O\mid q)]\frac{1}{|o|}\sum_{t=1}^{|o|}A_t\nabla_\theta\log\pi_\theta(o_t\mid q,o_{<t})$$

Data Source: question in SFT dataset with outputs sampled from policy model. Reward Function: reward model. Gradient Coefficient:

$$GC_{PPO}(q,o,t,\pi_{\theta_{rm}}) = A_t.$$

where $A\ t$ is the advantage, which is computed by applying Generalized Advantage Estimation (GAE) (Schulman et al., 2015), based on the rewards $\{r\geq t\}$ and a learned value function $V\ \psi$.

A.1.6. Group Relative Policy Optimization (GRPO)

The objective of GRPO is (assume $\pi_{\theta_{old}} = \pi_\theta$ for simplified analysis):

$$J_{GRPO}(\theta) = E[q\sim P_{sft}(Q),\{o_i\}_{i=1}^G\sim\pi_{\theta_{old}}(O\mid q)]\frac{1}{G}\sum_{i=1}^G\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left[\frac{\pi_\theta(o_{i,t}\mid q,o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}\mid q,o_{i,<t})}\hat{A}_{i,t}-\beta\left(\frac{\pi_{ref}(o_{i,t}\mid q,o_{i,<t})}{\pi_\theta(o_{i,t}\mid q,o_{i,<t})}-\log\frac{\pi_{ref}(o_{i,t}\mid q,o_{i,<t})}{\pi_\theta(o_{i,t}\mid q,o_{i,<t})}-1\right)\right].$$

The gradient of J *GRPO* (θ) is:

$$\nabla_\theta J_{GRPO}(\theta) = E[q\sim P_{sft}(Q),\{o_i\}_{i=1}^G\sim\pi_{\theta_{old}}(O\mid q)]\frac{1}{G}\sum_{i=1}^G\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left[\hat{A}_{i,t}+\beta\left(\frac{\pi_{ref}(o_{i,t}\mid o_{i,<t})}{\pi_\theta(o_{i,t}\mid o_{i,<t})}-1\right)\right]\nabla_\theta\log\pi_\theta(o_{i,t}\mid q,o_{i,<t}).$$

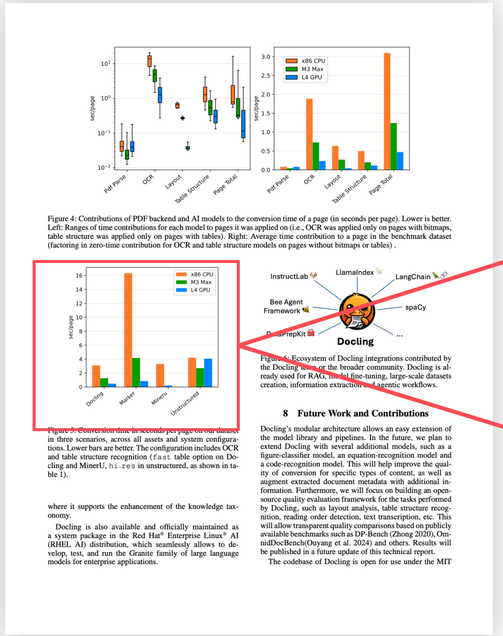
- Specialized enrichment for Code blocks and Formula
- Detection of 50+ programming languages
- Extraction of equation formula in LaTeX

https://huggingface.co/ds4sd/CodeFormula

Enrichment models

Picture understanding

```python
from docling.document_converter import DocumentConverter, PdfFormatOption
from docling.datamodel.pipeline_options import (
    PdfPipelineOptions, PictureDescriptionVlmOptions
)
from docling.datamodel.base_models import InputFormat

pipeline_options = PdfPipelineOptions()
pipeline_options.generate_picture_images = True
pipeline_options.images_scale = 2

# Picture classification
pipeline_options.do_picture_classification = True

# Picture description
pipeline_options.do_picture_description = True
picture_description_options = PictureDescriptionVlmOptions(
    repo_id="ibm-granite/granite-vision-3.2-2b",
    prompt="Describe the image in three sentences. Be consise and accurate.",
)

converter = DocumentConverter(format_options={
    InputFormat.PDF: PdfFormatOption(pipeline_options=pipeline_options)
})

result = converter.convert("https://arxiv.org/pdf/2501.17887")
doc = result.document
```




```
PictureClassificationClass(
    class_name='bar_chart',
    confidence=0.9999500513076782
)
```

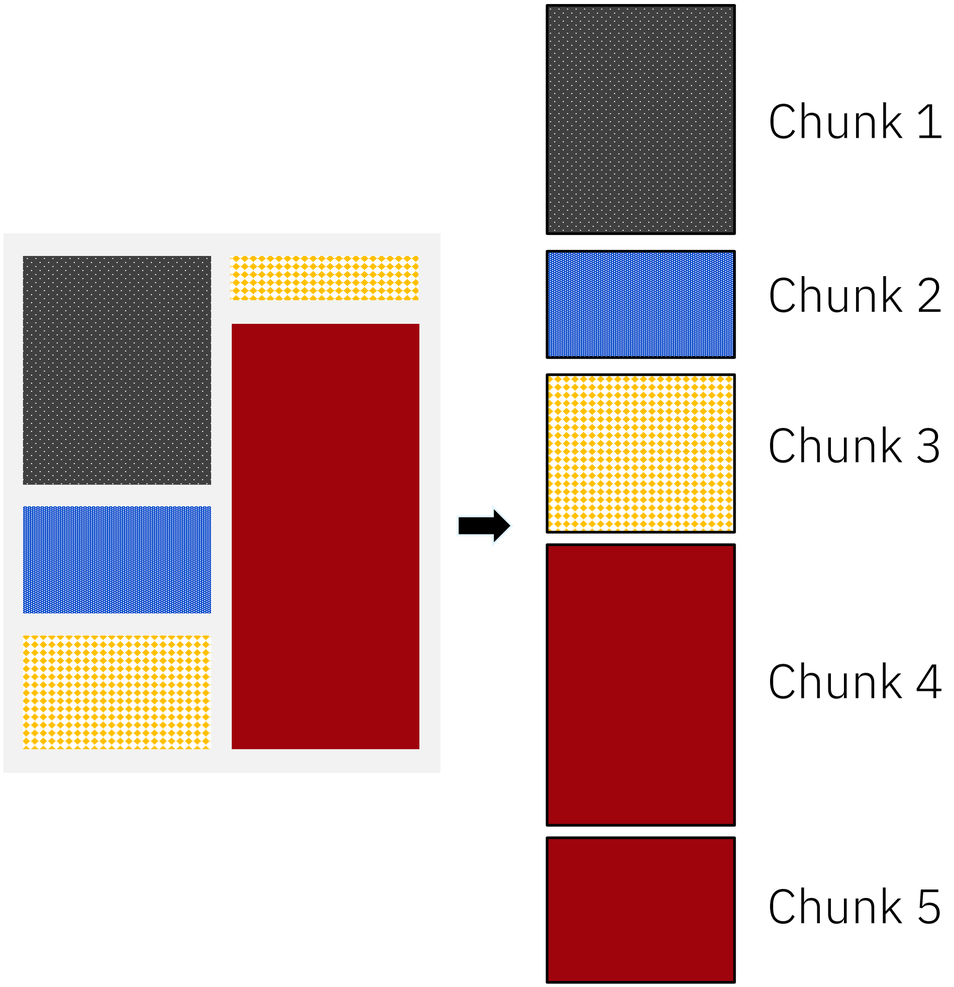
In this image we can see a bar chart. In the chart we can see the CPU, Max, GPU, and sec/page.

- Picture classifier specialized for figures found in documents, e.g. different chart types, flow diagrams, logos, signatures, etc.
- Leverage models to create a textual description of the picture

Chunking

- Docling supports different strategies via built-in chunkers & extensible design that allows plugging in user-defined ones
  - All native on rich `DoclingDocument` level, not just text
- Hierarchical: based on the doc layout extracted by Docling
- **Hybrid**: combines doc layout and tokenization awareness
- Any built-in or user-defined serializer can be used

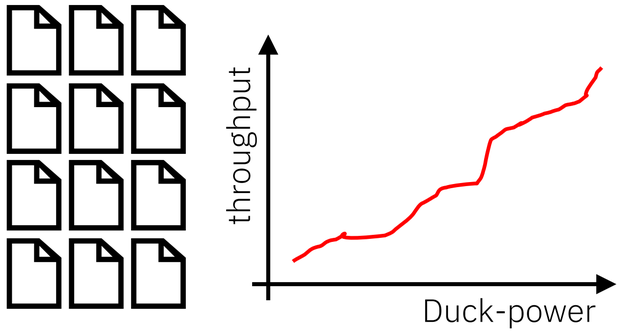
- Docs: https://docling-project.github.io/docling/concepts/chunking/
- Examples: https://docling-project.github.io/docling/examples/hybrid_chunking/



Hybrid chunking captures both document layout and token limitations

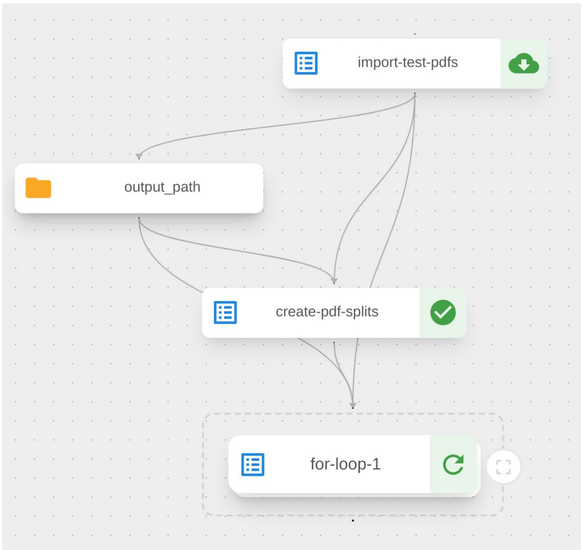
What is next?

Docling large scale jobs



Scale, scale, scale!

- Scale out to thousands of cores
- Support for multiple gpus
- Orchestrate conversion and enrichment steps (e.g. industry verticals)
- Telemetry and seamless monitoring of large jobs
- Built on OpenShift AI tools



Kubeflow

https://github.com/docling-project/docling-jobkit

IBM

Ecosystem

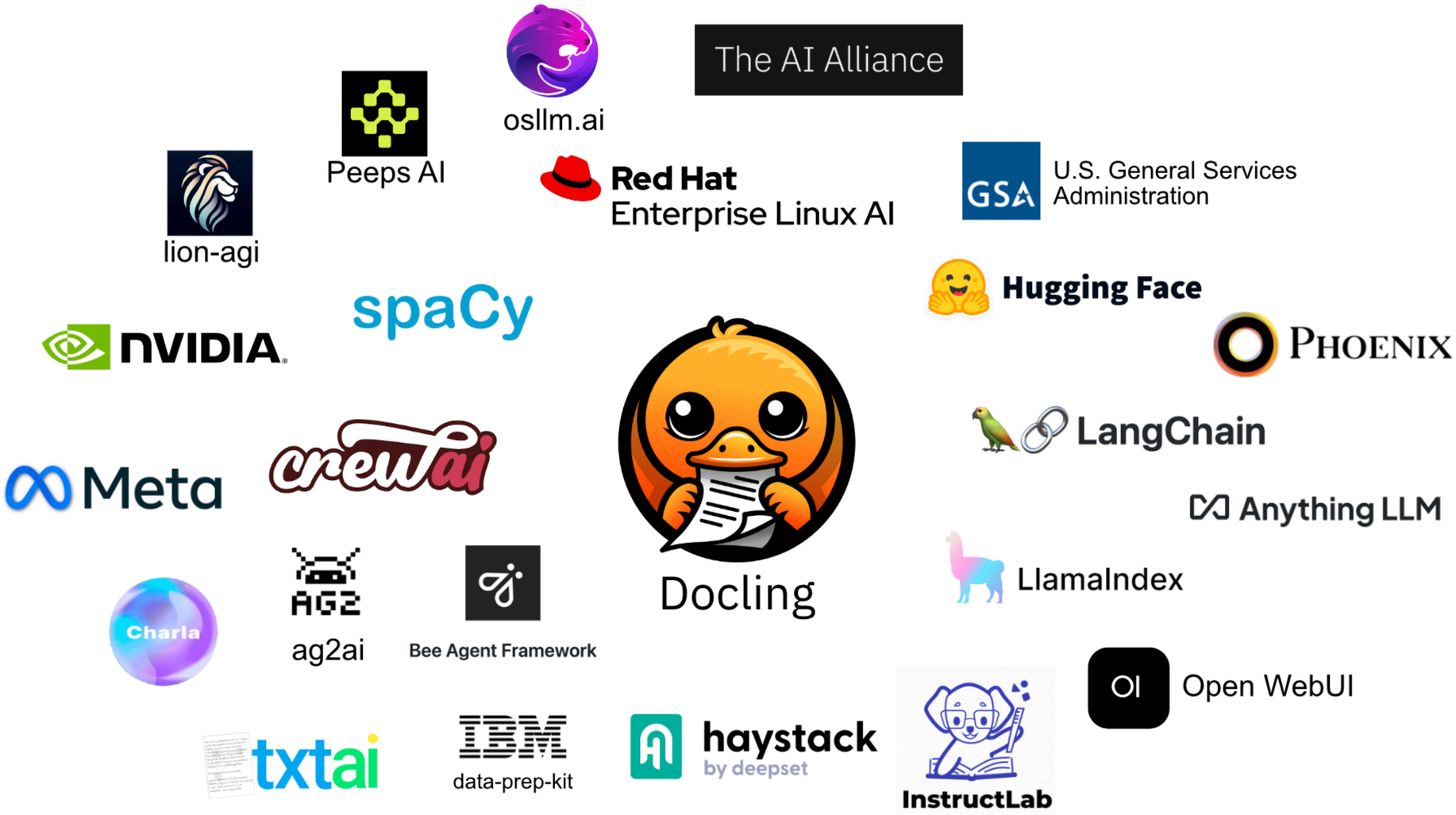
RAG Frameworks

Agentic systems

Dataset generation

Vector databases

Client solutions



IBM Research

Docling

Get your documents ready for gen AI

–

Ming Zhao
Software developer, Open Tech
mingzhao@IBM.com

https://www.linkedin.com/in/mingxuan-z-9a5a6419a/

https://github.com/mingxzhao/TechWeekCode

IBM