DocParser: Hierarchical Structure Parsing of Document Renderings

Johannes Rausch[1], Octavio Martinez[1], Fabian Bissig[1], Ce Zhang[1], and Stefan Feuerriegel[2]

[1]Department of Computer Science, ETH Zurich
[2]Department of Management, Technology, and Economics, ETH Zurich
johannes.rausch@inf.ethz.ch, octmb.94@gmail.com, fabian.bissig@alumni.ethz.ch,
ce.zhang@inf.ethz.ch, sfeuerriegel@ethz.ch

Abstract

Translating document renderings (e. g. PDFs, scans) into hierarchical structures is extensively demanded in the daily routines of many real-world applications, and is often a prerequisite step of many downstream NLP tasks. Earlier attempts focused on different but simpler tasks such as the detection of table or cell locations within documents; however, a holistic, principled approach to inferring the complete hierarchical structure in documents is missing. As a remedy, we developed "DocParser": an end-to-end system for parsing the complete document structure – including all text elements, figures, tables, and table cell structures. To the best of our knowledge, DocParser is the first system that derives the *full* hierarchical document compositions. Given the complexity of the task, annotating appropriate datasets is costly. Therefore, our second contribution is to provide a dataset for evaluating hierarchical document structure parsing. Our third contribution is to propose a scalable learning framework for settings where domain-specific data is scarce, which we address by a novel approach to weak supervision. Our computational experiments confirm the effectiveness of our proposed weak supervision: Compared to the baseline without weak supervision, it improves the mean average precision for detecting document entities by 37.1 %. When classifying hierarchical relations between entity pairs, it improves the F1 score by 21.2 %.

1 Introduction

Researchers, businesses, and organizations increasingly demand that documents are available in structured representations, as this is a crucial requirement in order to analyze their content automatically and often presents a prerequisite for downstream NLP tasks. However, such analyses are prevented by most file formats that are prevalent today due to being rendered without structural information. Prominent examples are PDF documents: this file format benefits from portability and immutability, yet it is flat in the sense that it stores all content as isolated entities (e. g., combinations of characters and positions) and, thus, hierarchical information is lacking. As such, the structure behind figures and especially tables is discarded and thus no longer available to computerized analyses. In contrast, file formats such as XML or JSON naturally encode hierarchical structures among textual entities. Hence, techniques are required in order to convert document renderings into structured, textual representations so that the textual content can be analyzed while considering the actual structure (Luong et al., 2012).

Earlier attempts for structure parsing on documents focused on a subset of simpler tasks such as segmentation of text regions (Antonacopoulos et al., 2009), locating tables (Zanibbi et al., 2004; Embley et al., 2006), or detecting hierarchical structures in tables (Schreiber et al., 2018). However, downstream tasks in NLP require representations of complete document structures.

A holistic, principled approach for inferring the complete hierarchical structure from documents is missing. On the one hand, such a task is non-trivial due to the complexity of documents, particularly their deeply-nested structures. For instance, nested tables are fairly easy to recognize for human readers, yet detecting them is known to impose computational hurdles (cf. Schreiber et al., 2018). On the other hand, efficient learning is prevented as large-scale training sets are lacking (cf. Arif and Shafait, 2018; Schreiber et al., 2018). Notably, prior datasets are limited to table structures (Gobel et al., 2013; Rice et al., 1995) and not the complete document structures. Needless to say, complex structures also make the labeling process significantly more costly (Wang et al., 2004). There-
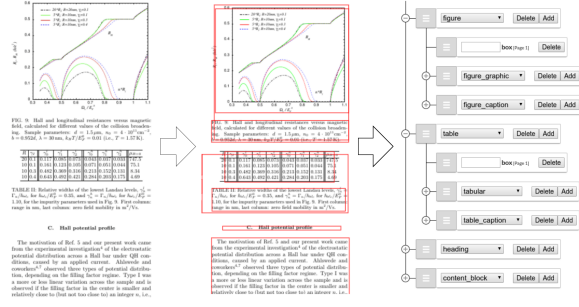
Figure 1: DocParser takes rendered document images (left) as input, performs segmentation into bounding boxes (center), and then outputs the hierarchical structure of the full document (right).

fore, an effective implementation that makes only a scarce use of labeled data is demanded.

This work focuses on parsing the hierarchical structure from renderings. We develop an end-to-end system for inferring the complete document structure (see Figure 1). This includes all entities (e. g., text, bibliography regions, figures, equations, headings, tables, and tables cells), as well as the hierarchical relations among them. We specifically adapt to settings in practice that suffer from data scarcity. For this purpose, we propose a novel learning framework for scalable weak supervision. It is intentionally tailored to the specific needs of parsing document renderings; that is, we create weakly-supervised labels by utilizing the reverse rendering process of LaTeX. The reverse rendering returns bounding boxes of all entities in documents together with their category (e. g., whether the entity is a table or a figure, etc.). Yet the outcomes are noisy (i. e., imprecise bounding boxes, missing entities, incorrect labels) and without deep structure information (e. g., information such as table row numbers is missing). Nevertheless, as we shall see later, the generated data greatly facilitates learning by being treated as weak labels.

Contributions:[1] We extend prior literature on document parsing in the following directions:

1. We contribute "DocParser". This presents the first end-to-end system for parsing document renderings into *hierarchical* structures. Prior literature has merely focused on simpler tasks such as table detection or table parsing but not the parsing of complete documents. As a remedy, we present a system for inferring document structures in a holistic, principled manner. For entity detection, our evaluations yield a mean average precision of 69.1; for classifying hierarchical relations between entities, we achieve an F1 score of 0.504.

2. We contribute the first dataset called "arXivdocs" for evaluating document parsing. It extends existing datasets for parsing in two directions: (i) it includes all entities that can appear in documents (i. e. not just tables) and (ii) it includes the hierarchical relations among them. The dataset is based on 127,472 scientific articles from the arXiv repository.

3. We propose a novel weakly-supervised learning framework. This is supposed to foster efficient learning in practice where annotated documents are scarce. It is based on an automated and thus scalable labeling process, where annotations are retrieved by reverse rendering the source code of documents. Specifically, in our work, we utilize TeXsource files from arXiv together with synctex for this objective. This then yields weakly-supervised labels by reverse rendering of the TeXcode.

4. We confirm the effectiveness of our weak supervision (based on the arXivdocs dataset): the mean average precision for entity detection is increased by 18.7 and the F1 score for classifying hierarchy relations by 0.088. This corresponds to a relative improvement of 37.1 % and 21.2 %, respectively. It also helps in outperforming the state-of-the-art performance on the related task of table parsing.

2 DocParser

2.1 Problem Description

Given a set of document renderings $D_1, \ldots, D_n$, the objective is to generate hierarchical structures $T_1, \ldots, T_n$. A hierarchical structure T_i, $i = 1, \ldots, n$, consists of both entities and relations as follows:

(i) **Entities** E_j, $j = 1, \ldots, m$, refer to the various elements within a document, such as, e. g. a figure, a table, a row, a cell, etc.[2] Each entity is described by three attributes: (1) its semantic category $c_j \in \mathcal{C} = \{C_1, \ldots, C_l\}$ (i. e., which defines the underlying type) and (2) the coordinates given by rectangular bounding box B_j on the document rendering. There is further (3) a confidence score P_j. This is not part of the

[2]For consistency, we use the term "entity" throughout the article in order to highlight its semantic nature. In computer vision, the term "object" is also common.

ground truth labels; however, comes with the predictions inside the DocParser system.

(ii) **Relations** R_j, $j = 1, \ldots, k$, are given by triples $(E_{\text{subj}}, E_{\text{obj}}, \Psi)$ consisting of a subject E_{subj}, an object E_{obj}, and a relation type $\Psi \in \{is\_parent\_of, is\_followed\_by, \text{null}\}$. The latter, null, is reserved for entities with meta-information that do not have designated order (i. e., header, footer, keywords, date, page number). All other entities must have $\Psi \neq \text{null}$.

The combination of entities and relations is sufficient to reconstruct the hierarchical structure T_i for a document. However, generating such a hierarchical structure from a document rendering D_i is subject to inherent challenges: the similar appearance of entities impedes detection and, further, the hierarchy can be nested arbitrarily, with substantial variation across different documents (e. g., some documents are single-, others multi-column).

2.2 System Overview

DocParser performs document structure parsing via 4 steps (as detailed in the following sections):

1. **Image conversion:** The document rendering D_i is converted into an image and further subject to preprocessing.

2. **Entity detection:** This step draws upon on a neural model (i. e., Mask R-CNN) for image segmentation and subsequent entity detection. Specifically, it takes the images from the previous step as input and then returns a flat list of entities $E_1, \ldots, E_m$ as output. For each, it comprises the semantic category, the coordinates in the form of bounding box, and a confidence score form the neural model. Our implementation makes use of 23 categories $\mathcal{C}$: content block, table, table row, table column, table cell, tabular, figure, heading, abstract, equation, itemize, item, bibliography block, table caption, figure graphic, figure caption, header, footer, page number, date, keywords, author, affiliation.

3. **Relation classification:** A set of heuristics is applied in order to translate the bounding boxes into hierarchical relations $R_1, \ldots, R_k$.

4. **Scalable Weak Supervision** The system is further extended by scalable weak supervision. The weak supervision aims at improving the performance of entity detection and, as a consequence, of end-to-end parsing.

In order to store document structures, we developed a customized, JSON-based file format.

2.3 System Components

2.3.1 Image Conversion

Document renderings are converted into images with a predefined resolution ρ. Further, all images are resized to a fixed rectangular size ϕ (if necessary, with zero padding).

The document images are further preprocessed: the RGB channels of all document images are normalized analogous to the MS COCO dataset (i. e., by subtracting inputs by the mean RGB channel values). The reason is that all neural models are later initialized with pre-trained weights from the MS COCO dataset (Lin et al., 2014).

2.3.2 Entity Detection

In order to detect all document entities within a document image, we build upon a neural model for image segmentation and subsequent entity detection, namely Mask R-CNN (He et al., 2017). For each entity, Mask R-CNN determines (i) its rectangular bounding box, (ii) a binary segmentation mask that distinguishes between the detected entity and background pixels within the bounding box, and (iii) a category label for the entity.

Mask R-CNN extends the architecture of a traditional convolution neural network (He et al., 2016) so that it is highly effective for image segmentation and object detection. Formally, it comprises of multiple stages with decreasing spatial resolution. The output of these stages is then fed into a so-called feature pyramid network (Lin et al., 2017) in order to produce multi-scale feature maps. The feature maps are then input to different prediction networks: first, a region proposal network generates a list of candidate bounding boxes that should contain an entity. Second, for each region proposal, a mask sub-network predicts the segmentation masks. These segmentation masks are not used in subsequent steps of DocParser; however, including them in the loss function stabilizes the overall learning process. Third, these bounding boxes are subsequently refined in a detection sub-network, thereby yielding the final bounding boxes B. It also provides the label for classifying the entity category.

All of the above sub-networks were carefully adapted to the specific characteristics of our task: (1) We modified the region proposal network so that it uses a maximum base aspect ratio of 1:8 per entity. This greatly improved the detection performance in DocParser. The reason for this mod-

ification is that document entities (as opposed to classical image segmentation) contain entities that have highly rectangular shapes. This is the case for most entities such as, e. g., single content lines, vertical text columns, or table rows. (2) The output size of the classifier sub-network is modified so that it can produce predictions for entities across all semantic categories $\mathcal{C}$. (3) During training of the mask sub-network, we treat all pixels in bounding boxes with ground truth as foreground. The reason is that many entities span very wide rectangular regions and it thus improves the detection performance for them. (4) We use a mask sub-network loss with a weighting factor of 0.5. This is supposed to prioritize that features relevant for the correct prediction of bounding boxes and entity categories are learned.

DocParser is built upon the implementation of Mask R-CNN provided by Abdulla (2017), yet which we carefully adapted as described above.

2.3.3 Relation Classification

Based on the flat list of entities from the previous step, a set of heuristics is utilized in order to retrieve the hierarchical structure. Here, we distinguish the heuristics according to whether they generate (1) the **nesting** among entities or (2) the **ordering** for entities of the same nesting level. The former case corresponds to $\Psi = is\_parent\_of$, while the latter determines all relations with $\Psi = is\_followed\_by$. Recall that we ignore all entities with meta-information (cf. Table 5), as these have no designated hierarchy.

Relations with nesting (is_parent_of): Four heuristics $h_1, \ldots, h_4$ determine parent-child relations, based on visual overlap of entities as well as a user-defined grammar. As a by-product, the heuristics for nesting also yield entities of the same nesting level. These represent siblings that are ordered as follows.

Relations with ordering ($is\_followed\_by$): The entities are ordered according to the general reading flow (i. e., from left to right). Here care is needed so that multi-column pages are processed correctly. For this, two heuristics o_1 and o_2 are used. The latter, o_2 is used for entities that are floating entities (i. e., figure, table), while o_1 is the default for all other entities.

Detailed descriptions of all heuristics used for relation classification are provided in the appendix.

2.3.4 Scalable Weak Supervision

Our weak supervision builds upon an additional dataset that consists of source codes (rather than document renderings). The source codes allow us to create a mapping between entities in the source code and their renderings. This has three particular characteristics: First, the mapping is noisy and thus created only weak labels. Despite that, the weak labels are supposed to aid efficient learning. Second, annotations are obtained only for some entities and relations. Third, if automated, this process circumvents human annotations and is thus highly scalable.

Let the unlabeled entities found in the source code be given by $S_1, \ldots, S_k$. For them, we generate weak labels $W_1, \ldots, W_k$ consisting of a semantic category and coordinates of the bounding box. However, both the semantic category and the bounding box can be subject to noise. Further, weak labels are generated merely for a subset $\mathcal{C}' \subseteq \mathcal{C}$ of the semantic categories.

In DocParser, the weak supervision is based on TEXsource files that are used to generate document renderings in the form of PDF files. The mapping between both formats is then obtained via `synctex` (Laurens, 2008). `synctex` is a synchronization tool that performs a reverse rendering, so that PDF locations are mapped to TEXcode. For given coordinates on the document rendering, `synctex` returns a list of rectangular bounding boxes and the corresponding source code. Notably, the inference bounding boxes represent noisy labels, since resulting entity annotations could be wrongly labeled, shifted, or entirely missing.

We proceed as follows. We iterate through the source code and retrieve bounding boxes for all TEXcommands. We then map the source code to our entities E. For instance, the bounding box for TEXcode `\includegraphics` inside a `\begin{figure}` ,..., `\end{figure}` environment is mapped onto an entity FIGURE_GRAPHIC[3] that is nested inside an entity FIGURE. A detailed description of all steps is given in Appendix D.
Bounding boxes for all entities that act as inner children are created dynamically by computing the union bounding of all child bounding boxes. During training, entities with obvious errors are dismissed, i. e., leaf nodes or generated entities with bounding boxes that extend beyond page limits or

---

[3]For consistency, this formatting is utilized for all entities.

with area 0.

2.4 System Variants

We compare the following variants of DocParser:

1. **DocParser Baseline** is trained solely on the noise-free labels provided for the training dataset (here: arXivdocs-target).

2. **DocParser WS** benefits from weak supervision (WS). It is trained based on a second dataset (here: arXivdocs-weak) with noisy labels for weak supervision. This is to test whether training systems on noisy labels can lead to higher performance, compared to training on small but noise-free training datasets.

3. **DocParser WS+FT** is initialized with the weights from DocParser WS, but it is then further fine-tuned (FT) on the target dataset.

3 Datasets with Document Structures

We contribute the dataset "arXivdocs" that is tailored to the task of hierarchical structure parsing. It comes in two variants: **arXivdocs-target** and **arXivdocs-weak**. (1) arXivdocs-target contains documents that have been manually checked and annotated. (2) arXivdocs-weak contains a large-scale set of documents that have no manual annotations but that can be used for weak supervision.

3.1 arXivdocs-target

arXivdocs-target provides a set of documents with manual annotations of the complete document structure. These documents were randomly selected from arXiv as an open repository of scientific articles, but in a way such that each has at most 30 pages and contains at least one table within the source code. Altogether, it counts 363 documents. arXivdocs-target comes with predefined splits for *training*, *validation* and *eval* that consist of 160, 79, 123 documents, respectively. The dataset comprises of 30 different entity categories.[4] We ensure a fairly uniform distribution of entity categories across different splits by sampling one random page rendering for each of the 362 documents that contain an abstract, figure, or table. The *eval* split is used to measure the out-of-sample performance.

Descriptive statistics of the dataset are as follows. On average, each document contains 86.32 entities. Figure 2 shows the number of leaf nodes in the

---

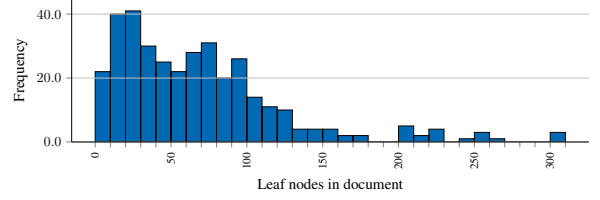[4]Some entity categories are extremely rare and, hence, only a subset is later used as part of our experiments.

Figure 2: Number of leaf nodes in documents.

Category	Frequency	%	Avg. depth
abstract	63	0.20	1.00
affiliation	82	0.26	1.00
author	89	0.28	1.00
bibliogr. block	32	0.10	2.00
bibliography	24	0.08	1.08
code	3	0.01	2.00
content block	1009	3.23	2.05
content line	10,729	34.33	3.06
content lines	627	2.01	3.04
date	16	0.05	1.00
equation	353	1.13	1.97
equation formula	364	1.16	3.01
equation label	275	0.88	2.95
figure	607	1.94	2.44
figure caption	404	1.29	3.16
figure graphic	454	1.45	3.63
footer	81	0.26	1.00
header	106	0.34	1.01
heading	398	1.27	2.10
item	63	0.20	3.08
itemize	24	0.08	1.96
page nr	261	0.84	1.00
section	527	1.69	1.29
keywords	36	0.12	1.00
table	185	0.59	1.81
table caption	183	0.59	2.80
table cell	11,146	35.67	3.54
table col	1109	3.55	3.69
table row	1812	5.80	3.68
tabular	187	0.60	2.80

Table 1: Statistics by entity of arXivdocs-target.

document graph. In addition, Figure 1 reports the frequency and the average depth of the different entities. Evidently, the most common category in the dataset is content line (34.33 %). Content lines are especially frequent as they typically represent leaf nodes in the graph and are children of larger entities, such as abstract, captions, or text blocks.

3.2 arXivdocs-weak

arXivdocs-weak contains 127,472 documents with an average length of 12.84 pages that were retrieved from arXiv. We selected only documents that have a length of at most 30 pages and contain at least one table within their source code. For reproducibility, we make our weak labels available.[5]

4 Computational Setup

4.1 Training Details

Training procedure: All neural models are initialized with pre-trained weights from the MS COCO dataset (Lin et al., 2014). We then train each model

---

[5]For this purpose, the dataset was labeled via our proposed weak supervision mechanism and thus contains both entities E_j and hierarchical relations R_j. For reasons of space of the physical files, bounding boxes are only stored for entities in leaf nodes. For all other entities, the bounding boxes can be calculated by taking the union bounding box of their children.

across three phases for a total of up to 80,000 iterations.

Weak supervision: For training with weak supervision, all models are initialized with the weights of our pre-trained DocParser WS instead of default weights.

Further details about training procedure, parameter settings and weak supervision can be found in the appendix.

4.2 Performance Metrics

Document structure parsing is evaluated based on two performance metrics with respect to (i) entity detection E_j and (ii) classification of hierarchical relations R_j as follows.

Entity detection: The inferred entity $E_j = (c_j, B_j, P_j)$ is compared against the ground truth label consisting of the true category $\hat{c}_j$ with a bounding box $\hat{B}_j$. Here we follow common practice in computer vision (Everingham et al., 2010) and measure the overlap between bounding boxes from the same category. Specifically, we calculate the so-called intersection over union (IoU):

$$\text{IoU} = \frac{area(B_j \cap \hat{B}_j)}{area(B_j \cup \hat{B}_j)}. \tag{1}$$

If the IoU is higher than a user-defined threshold, a predicted entity is considered a true positive. If multiple entities are matched with the same ground truth entity, we only consider the predicted with the highest IoU as a true positive. Unmatched predictions and ground truth entities are considered false positives and false negatives, respectively. This is then used to calculate the average precision (AP) per semantic category $C_k \in \mathcal{C}$. The overall performance across all categories is given by the mean average precision (mAP). In our experiments, we compare IoU thresholds of 0.5, 0.65, and 0.8.

Prediction of hierarchical relations: Here we measure the classification performance for predicting the correct relations. A relation $R = (E_{\text{subj}}, E_{\text{obj}}, \Psi)$ is counted as correct iff. the complete tuple is identical. However, the performance depends on the correct entity detection as input. Hence, we later vary the IoU thresholds for entity detection analogous to above and then report the corresponding F1 score for correctly predicting hierarchical relations.

5 Results

The key focus of our experiments is to confirm the effectiveness of DocParser for parsing the com-

AP	IoU=0.5			IoU=0.65			IoU=0.8		
	Baseline	WS	WS+FT	Baseline	WS	WS+FT	Baseline	WS	WS+FT
mAP	50.4	35.6	**69.1**	38.1	34.5	**56.4**	14.2	22.6	**33.1**
abstract	**95.2**	90.5	95.2	90.5	84.5	**95.2**	56.7	26.6	**75.1**
affiliation	**57.9**	0.0	47.7	5.6	0.0	**16.4**	**1.0**	0.0	0.1
author	18.0	0.0	**25.9**	**19.4**	0.0	17.1	4.9	0.0	**5.7**
bib. block	42.4	79.1	**94.7**	43.2	**93.9**	80.3	13.6	**96.2**	71.6
cont. block	88.7	74.0	**90.2**	82.9	70.0	**87.2**	64.0	58.5	**75.3**
date	0.0	0.0	**24.1**	0.0	0.0	**9.3**	**0.0**	0.0	0.0
equation	65.7	55.9	**81.9**	41.3	53.2	**72.8**	8.9	**37.4**	36.8
fig. caption	48.5	30.5	**69.5**	44.7	17.7	**59.5**	17.3	19.6	**39.8**
fig. graphic	22.6	5.2	**60.6**	16.6	4.4	**55.0**	6.0	1.6	**36.5**
figure	**49.1**	33.5	44.0	**43.5**	33.1	39.6	13.1	12.2	**36.4**
footer	55.7	0.0	**69.3**	48.9	0.0	**59.7**	5.0	0.0	**7.9**
header	79.7	0.0	**88.3**	**64.8**	0.0	56.6	**12.1**	0.0	6.5
heading	54.5	52.1	**70.7**	33.3	46.7	**54.2**	6.7	**26.3**	24.5
item	0.0	32.2	**52.6**	0.0	**40.0**	36.8	0.0	**53.0**	13.6
itemize	0.0	41.7	**58.3**	0.0	**50.0**	50.0	0.0	0.0	**20.8**
keywords	36.4	0.0	**59.0**	36.4	0.0	**43.0**	20.5	0.0	**22.3**
page nr.	74.7	0.0	**77.3**	28.5	0.0	**42.0**	0.8	0.0	**2.0**
tab. caption	55.2	69.1	**78.7**	40.1	63.0	**63.9**	16.5	**41.5**	28.5
table	86.1	**96.3**	95.1	52.0	**91.9**	90.4	6.4	57.7	**69.9**
tabular	78.4	50.8	**100.0**	69.4	42.4	**99.5**	30.0	22.3	**89.0**

Table 2: Comparison of entity detection (average precision).

plete document structures. However, we emphasize again that both suitable baselines and datasets for this task are hitherto lacking. Hence, we proceed two-fold. On the one hand, we evaluate the performance based on arXivdocs as the first dataset for document structure parsing. On the other hand, we draw upon the table structure ICDAR 2013 dataset: it is limited to table structures and not complete holistic parsing of document structures. However, it allows to test the effectiveness of our weak supervision against state-of-the-art.

5.1 Performance of Document Structure Parsing

We compare the performance of document structure parsing based on our arXivdocs-target dataset across both performance metrics.

5.1.1 Entity Detection

The overall performance for entity detection is detailed in Table 2 (first row). For IoU = 0.5, the default DocParser achieves a mAP of 50.4. This is higher than DocParser WS with a mAP of 35.6. We attribute this to the fact that several entity categories from arXivdocs-target are not part of arXivdocs-weak. Notably, the fine-tuned system DocParser WS+FT results in significant performance improvements: it obtains a mAP of 69.1, which, in comparison to the default DocParser, is an improvement by 37.1 %. On top of that, DocParser WS+FT outperforms consistently the default DocParser system across all measured IoU thresholds. Using IoU thresholds above 0.5 leads to a performance decrease. Even though higher IoUs should generally correspond to better matches with the ground truth, they can penalize ambiguous

cases and thus a correct detection. In sum, this confirms the effectiveness of our weak supervision in bolstering the overall performance.

Table 2 entails a breakdown of performance by entity category. For DocParser WS+FT, we observe an especially good performance for detecting tables (i. e., TABLE and TABULAR categories). This is owed to our strong initialization of our system due to the high quality and the large number of samples in our scalable weak supervision.[6]
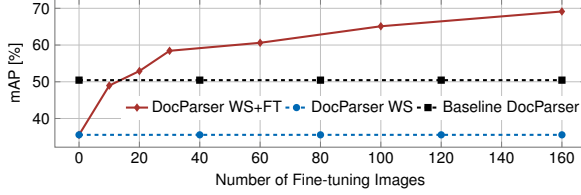

Figure 3: Performance of entity detection (mAP for $IoU = 0.5$) during fine-tuning.

Figure 3 shows the fine-tuning. Only 20 fine-tuning samples are sufficient for DocParser WS+FT to surpass the default system DocParser (which is trained on 160 samples from the target dataset)[7]. It thus helps in reducing the labeling effort by a factor of around 8. Further, we observe a steady increase in performance of the fine-tuned networks with more samples. Notably, the highest performance increase is already achieved by the first 10 document images for fine-tuning.

5.1.2 Prediction of Hierarchical Relations

	IoU=0.5			IoU=0.65			IoU=0.8		
	Baseline	WS	WS+FT	Baseline	WS	WS+FT	Baseline	WS	WS+FT
All	0.416	0.343	**0.504**	0.322	0.318	**0.445**	0.114	0.214	**0.313**
is_followed_by	0.413	0.387	**0.506**	0.314	0.366	**0.447**	0.118	0.244	**0.308**
is_parent_of	0.421	0.235	**0.500**	0.339	0.198	**0.443**	0.103	0.138	**0.325**

Table 3: Performance in predicting hierarchical relations (as measured by F1).

Table 3 compares the classification of hierarchy relations. The best performance (across all Ψ) is achieved by DocParser WS+FT with an IoU of 0.5: it registers an F1 score of 0.504. Here the use of weak supervision with fine-tuning yields

---

[6]For a few entities, the best performance is achieved a combination of the WS system together with a high IoU (e. g., BIBLIOGRAPHY BLOCK). A likely reason for this is the composition of arXivdocs-target. As bibliography entities were not specifically used as a criterion for the per-page sampling, fewer documents in the target dataset contained relevant entities, leading to decreased performance of the default DocParser and DocParser WS+FT systems.

[7]We report average scores over three runs with differing random seeds for all fine-tuning experiments on arXivdocs-target to account for the small number of training samples

System	F1*	F1
DocParser Baseline	0.8443	0.8209
DocParser WS	0.8117	0.8056
DocParser WS+FT	**0.9292**	**0.9292**
Schreiber et al. (2018)	0.9144	—

Table 4: Effectiveness of our weak supervision for table structure parsing (based on ICDAR 2013).

Notes: Evaluation of image-based systems on "ICDAR 50 %", which uses a random subset containing 50 % of the competition set for testing. (Schreiber et al., 2018) uses a different, non-public 50 % random subset. Furthermore, (Schreiber et al., 2018) chooses the best system based on the test set as indicated by F1*. In contrast, F1 refers to the performance when the selection is based on the validation set.

consistent improvements. In particular, for an IoU of 0.5, it outperforms the F1 score of the default system (F1 of 0.416) by 0.088. This amounts to a relative improvement of 21.2 %. Again a smaller IoU threshold of 0.5 seems beneficial. As structure parsing builds on the prior detection of document entities, higher IoU thresholds further reduce the overall parsing performance.

5.2 Robustness Check: Table Structure Parsing

We confirm the effectiveness of our weak supervision as follows: we draw upon the ICDAR 2013 dataset (Gobel et al., 2013) for table structure parsing and compare it against state-of-the-art systems for this task.

Setting: We now train models for structure parsing so that it identifies table structures. All experiments use the identical setups as in our default training procedures. Fine-tuning follows the 3 phase training scheme for a total of 80,000 iterations. The ICDAR labels now serve as the target dataset, while we continue to use arXivdocs-weak for weak supervision.[8] Following (Schreiber et al., 2018), we use a random subset of 50 % of the ICDAR 2013 competition dataset for testing. The remaining competition samples are mixed with the practice samples in order to generate the training and validation sets.

We match our table cell predictions with the text element locations provided by (Nurminen, 2013) in order to generate XML files that are compared to the ground truth by the scripts provided on the competition website. Matches are determined by the fraction of overlap between cell and text bounding boxes $\gamma = \frac{area(B_{\text{cell}} \cap B_{\text{text}})}{area(B_{\text{text}})}$, using $\gamma \geq 0.5$.

Results: Table 4 compares state-of-the-art deep learning for table structure parsing against our pro-

---

[8]Due to the different domain of the target dataset, we experimented with other weak supervision strategies, e. g. randomly sampling images from arXivdocs-weak and ICDAR 2013 during the same training procedure. However, the performance of models trained by sequential fine-tuning could not be surpassed.

posed weak supervision strategy. Altogether, our weak supervision outperforms the state-of-the-art with image-based input (Schreiber et al., 2018) by a considerable margin.

Discussion: Our system shows significant improvement over the image-based state of the art. We also compare our approach to the state-of-the-art heuristic-based system that operates on raw PDF files, instead of images, as input (Nurminen, 2013). Even though our system does not utilize the additional information provided by raw PDF files, Doc-Parser achieves an F1 score of 0.9292, compared to 0.9221 for the PDF-based system.[9]

6 Related Work

OCR: Extracting text from document images has been extensively studied as part of optical character recognition (OCR) within the NLP community (e.g., Schäfer et al., 2011; Schäfer and Weitz, 2012). To this end, the work by Katti et al. (2018) argued that OCR should be seen as a preprocessing step for downstream NLP tasks. As such, the authors extract text-based information but not the hierarchical document structure as in our research.

Table detection: Document renderings are commonly used for the task of table detection (rather than table structure parsing). Here the objective is simply to predict the bounding boxes of tables, i.e., whether a pixel refers to a table or not. This task was earlier approached with rule-based mechanisms that adapt to specific structural elements (e.g., Yildiz et al., 2005). In most cases, these rules were making use of additional PDF metadata, e.g., print instructions, line segments, or actual character bounding boxes. Table detection has also been addressed by probabilistic classifiers (Wang et al., 2004), ensembles (Fan and Kim, 2015), and even tailored neural classifiers for image segmentation (Li et al., 2018). The latter, neural classifiers for image segmentation, commonly draws upon Mask R-CNN or computationally-efficient variants of it such as, for instance, Fast R-CNN as in (Siddiqui et al., 2018) or Faster R-CNN as in (Arif and Shafait, 2018; Gilani et al., 2017). These variants of Mask R-CNN differ in their computational speed-up while obtaining a similar performance.

Efficient learning presents an issue for table detection to a similar extent as it is for our research.

Prior research on table detection has utilized distant supervision via manual annotation rules (Fan and Kim, 2015), data augmentation (Gilani et al., 2017) and transfer learning (e.g., Siddiqui et al., 2018) to address the lack of large-scale domain-specific datasets, but not weak supervision.

Extensions have been developed in order to adapt table detection to general entity detection (e.g., Li et al., 2018; Wang et al., 2004) in document renderings, where, e.g., also figures or formulae must be detected. However, a wide variety of entities and their hierarchical relations are not part of this task.

Table structure parsing: There are works that recognize table structures from text or other syntactic tokens (Kieninger and Dengel, 1998, 2001; Pivk et al., 2007; Shigarov et al., 2016) rather than directly from document renderings. As such, these works are tailored to tokens as input, and it is thus unclear how such an approach could theoretically be adapted to document renderings since our task inherently relies upon images as input. Because of the different input and thus the different datasets for benchmarking, the performance of the aforementioned works is not comparable to our approach. The work by Schreiber et al. (2018) draws upon a CNN to perform semantic segmentation of row and column pixels and identifies structure via post-processing. However, it aims at a different purpose: parsing table structures, but not complete document hierarchies. As such, the authors do not attempt to identify text elements, figures, etc.

Weak Supervision in NLP: Annotations in NLP are oftentimes costly and, as a result, there has been a recent surge in weak supervision. Weak supervision has now been applied to various tasks, such as text classification (e.g., Hingmire and Chakraborti, 2014; Lin et al., 2011), information extraction (e.g., Hoffmann et al., 2011), and semantic parsing (e.g., Goldman et al., 2018). The methodological levers for obtaining weak labels are versatile and include, e.g., manual rules (e.g., Rabinovich et al., 2018), estimated models (e.g., Hoffmann et al., 2011) or reinforcement learning (Pröllochs et al., 2019); however, not for document structure parsing.

References

Waleed Abdulla. 2017. Mask R-CNN for Object Detection and Instance Segmentation on Keras and TensorFlow. https://github.com/matterport/Mask_RCNN.

---

[9] We received the outputs for the ICDAR "competition" dataset from the authors of (Nurminen, 2013). We used the evaluation script provided by the competition organizers to calculate the ICDAR 50% performance.

Apostolos Antonacopoulos, David Bridson, Christos Papadopoulos, and Stefan Pletschacher. 2009. A Realistic Dataset for Performance Evaluation of Document Layout Analysis. In *International Conference on Document Analysis and Recognition (ICDAR)*.

Saman Arif and Faisal Shafait. 2018. Table Detection in Document Images using Foreground and Background Features. In *2018 Digital Image Computing: Techniques and Applications (DICTA)*.

David W. Embley, Matthew Hurst, Daniel Lopresti, and George Nagy. 2006. Table-processing Paradigms: A Research Survey.

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. 2010. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338.

Miao Fan and Doo Soon Kim. 2015. Detecting Table Region in PDF Documents Using Distant Supervision. *arXiv:1506.08891*.

Azka Gilani, Shah Rukh Qasim, Imran Malik, and Faisal Shafait. 2017. Table detection using deep learning. In *14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*.

Max Gobel, Tamir Hassan, Ermelinda Oro, and Giorgio Orsi. 2013. ICDAR 2013 Table Competition. In *International Conference on Document Analysis and Recognition (ICDAR)*.

Omer Goldman, Veronica Latcinnik, Ehud Nave, Amir Globerson, and Jonathan Berant. 2018. Weakly Supervised Semantic Parsing with Abstract Examples. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Swapnil Hingmire and Sutanu Chakraborti. 2014. Sprinkling Topics For Weakly Supervised Text Classification. In *Annual Meeting of the ACL*.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based Weak Supervision for Information Extraction of Overlapping Relations. In *Annual Meeting of the ACL*.

Anoop R Katti, Christian Reisswig, Cordula Guder, Sebastian Brarda, Steffen Bickel, Johannes Höhne, and Jean Baptiste Faddoul. 2018. Chargrid: Towards understanding 2d documents. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Thomas Kieninger and Andreas Dengel. 1998. The t-recs table recognition and analysis system. In *International Workshop on Document Analysis Systems (DAS)*.

Thomas Kieninger and Andreas Dengel. 2001. Applying the T-Recs Table Recognition System to the Business Letter Domain. In *International Conference on Document Analysis and Recognition (ICDAR)*.

Jerôme Laurens. 2008. Direct and reverse synchronization with SyncTEX. *TUGBoat*, 29:365–371.

Xiao Hui Li, Fei Yin, and Cheng Lin Liu. 2018. Page Object Detection from PDF Document Images by Deep Structured Prediction and Supervised Clustering. In *Proceedings - International Conference on Pattern Recognition*.

Chenghua Lin, Yulan He, and and Everson Richard. 2011. Sentence Subjectivity Detection With Weakly-Supervised Learning. In *International Joint Conference on Natural Language Processing (IJCNLP)*.

Tsung Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. 2017. Feature Pyramid Networks for Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Tsung Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision (ECCV)*.

Minh-Thang Luong, Thuy Dung Nguyen, and Min-Yen Kan. 2012. Logical structure recovery in scholarly articles with rich document features. In *Multimedia Storage and Retrieval Innovations for Digital Library Systems*, pages 270–292. IGI Global.

Anssi Nurminen. 2013. Algorithmic extraction of data in tables in pdf documents. Master's thesis, Tampere University of Technology.

Aleksander Pivk, Philipp Cimiano, York Sure, Matjaz Gams, Vladislav Rajkovič, and Rudi Studer. 2007. Transforming Arbitrary Tables into Logical Form with TARTAR. *Data and Knowledge Engineering*, pages 567–595.

Nicolas Pröllochs, Stefan Feuerriegel, and Dirk Neumann. 2019. Learning Interpretable Negation Rules via Weak Supervision at Document Level: A Reinforcement Learning Approach. In *NAACL-HLT*.

Ella Rabinovich, Benjamin Sznajder, Artem Spector, Ilya Shnayderman, Ranit Aharonov, David Konopnicki, and Noam Slonim. 2018. Learning Concept Abstractness using Weak Supervision. In *EMNLP*.

Stephen V Rice, Frank R Jenkins, and Thomas A Nartker. 1995. The fourth annual test of ocr accuracy. Technical report, Technical Report 95.

Ulrich Schäfer, Bernd Kiefer, Christian Spurk, Jörg Steffen, and Rui Wang. 2011. The acl anthology searchbench. In *49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Systems Demonstrations (ACL-HLT)*. Association for Computational Linguistics.

Ulrich Schäfer and Benjamin Weitz. 2012. Combining ocr outputs for logical document structure markup: Technical background to the acl 2012 contributed task. In *ACL-2012 Special Workshop on Rediscovering 50 Years of Discoveries*, ACL '12.

Sebastian Schreiber, Stefan Agne, Ivo Wolf, Andreas Dengel, and Sheraz Ahmed. 2018. DeepDeSRT: Deep Learning for Detection and Structure Recognition of Tables in Document Images. In *International Conference on Document Analysis and Recognition (ICDAR)*.

Alexey Shigarov, Andrey Mikhailov, and Andrey Altaev. 2016. Configurable table structure recognition in untagged pdf documents. In *2016 ACM Symposium on Document Engineering (DocEng)*. ACM.

Shoaib Ahmed Siddiqui, Muhammad Imran Malik, Stefan Agne, Andreas Dengel, and Sheraz Ahmed. 2018. DeCNT: Deep Deformable CNN for Table Detection. *IEEE Access*, pages 74151–74161.

Yalin Wang, Ihsin T. Phillips, and Robert M. Haralick. 2004. Table Structure Understanding and its Performance Evaluation. *Pattern Recognition*, pages 1479–1497.

Burcu Yildiz, Katharina Kaiser, and Silvia Miksch. 2005. pdf2table: A Method to Extract Table Information from PDF Files. *2nd Indian International Conference on Artificial Intelligence (IICAI)*.

Richard Zanibbi, Dorothea Blostein, and JamesR. Cordy. 2004. A Survey of Table Recognition. *Document Analysis and Recognition*, pages 1–33.

Entity ($\mathcal{C}$)	Relation types Ψ	Valid entities	Notes
Abstract	*is_parent_of*	Heading	
Date	null	—	Meta
Figure	*is_parent_of*	Figure,	Float
		Fig. graphic,	
		Fig. caption,	
Footer	null	—	Meta
Header	null	—	Meta
Item	*is_parent_of*	Equation	
Itemize	*is_parent_of*	Item	
Keywords	null	—	Meta
Page nr	null	—	Meta
Table	*is_parent_of*	Tabular	Float
		Table caption,	
All others	*is_parent_of*,	—	
	is_followed_by	*any sibling*	

Table 5: Document grammar for different entity categories that is utilized in our heuristics. Every category can by default exist on the highest hierarchical level, i. e., without being nested.
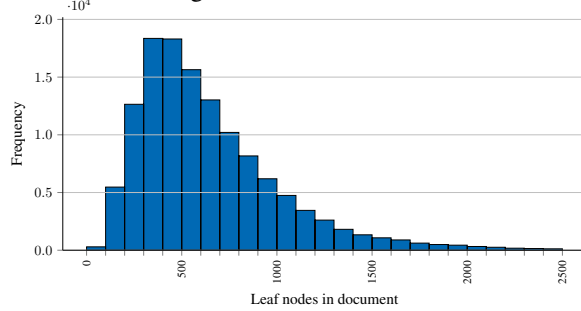

Figure 4: Total number of leaf nodes in the hierarchical structure graph per document.

Appendix A Document Grammar

Hierarchical relations between entity pairs follow a predefined grammar (see Table 5). All entities with meta-information have no ordering, i. e., their relation type is $\Psi = $ null. Some entities (such as, e. g., figures) have only a certain set of allowed child entities. For instance, a figure can contain a figure caption a graphic, or a subfigure (i. e., another nested figure), but not other entities such as a table or an abstract. Finally, the hierarchical structures T_i must form a tree. That is, an entity is allowed to have multiple ordered siblings (i. e., multiple entities with the same nesting level). However, each entity must only have one parent, i. e., for an entity E there is exactly one relation (E', E, is_parent_of) with an entity $E \neq E$.

Appendix B Datasets with Document Structure: arXivdocs-weak

Descriptive statistics of the dataset are as follows. Figure 4 reports the number of leaf nodes in the document graph. Table 6 shows the frequency and the average depth of the different entities. Evidently, the most common category in the dataset is content line. Content lines typically represent leaf nodes in the graph and are children of larger

Category	Frequency	%	Avg. Depth
abstract	89,291	0.09	1.00
author	48	0.00	3.00
bibliogr. block	242,412	0.26	2.94
bibliography	80,864	0.09	1.93
caption	26	0.00	4.38
content block	5,033,714	5.32	2.41
content line	63,339,623	66.96	3.47
date	5	0.00	2.00
equation	1,489,078	1.57	2.44
equation formula	1,743,705	1.84	3.43
equation label	1,503,778	1.59	3.44
figure	478,086	0.51	2.50
figure caption	263,495	0.28	3.48
figure graphic	408,088	0.43	3.52
heading	975,414	1.03	2.54
item	436,222	0.46	3.58
itemize	140,415	0.15	2.64
meta	127,477	0.13	3.00
section	1,296,707	1.37	1.56
table	292,110	0.31	2.43
table caption	206,215	0.22	3.42
table cell	12,343,327	13.05	4.40
table col	1,285,945	1.36	4.42
table row	2,533,799	2.68	4.41
tabular	280,572	0.30	3.43
title	16	0.00	3.00

Table 6: Summary statistics by entity of arXivdocs-weak dataset.

entities, such as abstract, captions, or text blocks.

Appendix C Relation Classification

Relations with nesting (*is_parent_of*): Four heuristics $h_1, \ldots, h_4$ determine parent-child relationn as follows:

h_1 **Overlaps.** A list of candidate parent-child relations is compiled based on the overlap of bounding boxes as follows. That is, DocParser loops over all bounding boxes and, for each bounding box B_{subj}, it determines all other bounding boxes that are contained within B_{subj}. Formally, this is given by all tuples of bounding boxes $(B_{\text{subj}}, B_{\text{obj}})$ with subj $\in m$, obj $\in m$, and subj $\neq$ obj where $h_1(B_{\text{subj}}, B_{\text{obj}})$ is satisfied: Tuples for which all 4 coordinates of B_{obj} are enclosed by the bounding rectangle B_{subj} are added to the candidate list. Furthermore, we add tuples to the candidate list that satisfy $\frac{area(B_{\text{subj}} \cap B_{\text{obj}})}{area(B_{\text{obj}})} \geq \theta_1$ and $\frac{area(B_{\text{subj}})}{area(B_{\text{obj}})} > \theta_2$, i. e. they must have a certain overlap fraction θ_1 and size ratio θ_2. In DocParser, thresholds of $\theta_1 = 0.45$ and $\theta_2 = 1.2$ are used.

h_2 **Grammar check.** This heuristic validates the candidate list against the predefined document grammar (see Table 5). Concretely, all illegal candidates are removed.

h_3 **Direct children.** The candidate list is further pruned so that it contains only direct children of the parent and not sub-children. For this purpose, all sub-children

are removed. As an example, this should remove $(E_{\text{subj}}^1, E_{\text{obj}}^3)$ from a candidate list $\{(E_{\text{subj}}^1, E_{\text{obj}}^2), (E_{\text{subj}}^1, E_{\text{obj}}^3), (E_{\text{subj}}^2, E_{\text{obj}}^3)\}$, since it represents a sub-child and not a direct child of E_{subj}.

h_4 **Unique Parents.** The candidate list is further altered so that each entity has only a single parent. Formally, if an entity E_{obj} has multiple candidate parents, we first compare the IoU of the bounding boxes of all candidate parents with E_{obj}. We then keep the parent with the maximal IoU, while all others are removed. If two parents have the same IoU, we select the element with the highest confidence score P_j as parent. If that value is also equal, we choose the entity with the largest bounding box.

Relations with ordering (*is_followed_by*): The two heuristics o_1 and o_2 are used to identify relations with ordering as follows:

o_1 **Non-floating entities.** First, all entities are grouped according to their coordinates on the document pages, namely into groups belonging to the (a) left side G_l, (b) center G_c, or (c) right side G_r. Formally, this is achieved by computing the overlap for each entity E_j, $j = 1, \ldots, m$ with the left (and right) side of a document page, i.e., $\tau_{\text{ovlp}} = \text{overlap}/\textit{width}(B)$. If the overlap with either the left (or the right) side is above a above threshold (i.e., $\tau_{\text{ovlp}} > 0.7$), the entity E_j is assigned to the left (or right) side. Otherwise, if such assignment is not possible with high confidence, the entity E_j is assigned to center group G_c. In other words, the center group should collect all entities such as content lines spanning the complete page width. However, in a two-column format, all entities are assigned to the left and right group only. In essence, the center group is an indicator whether the document is in single- or multi-column. These groupings are then proceeded further by the o_2 heuristic.[10]

---

[10]If no entities have been assigned to the center group (i.e., $G_c = \emptyset$), then the entities are ordered first according to G_l followed by G_r. Within each group, the entities are ordered top-to-bottom and then left-to-right by applying heuristic o_2. In sum, this approach should find an appropriate ordering for multi-column pages.

If entities have been assigned to the center group (i.e., $G_c \neq \emptyset$), then grouping is further decomposed into additional subgroups: the entities $E \in G_c$ from the center group are used to split G_l, G_c, and G_r into vertical subgroups G_l^ι, G_c^ι, and G_r^ι, respectively. Afterward, we loop over all vertical subgroups ι. For each, we order the entities according to the group (first G_l^ι, followed by G_c^ι and then G_r^ι). Within each subgroup, we perform the ordering via heuristic o_2. This

o_2 **Floating entities.** The entities E_j, $j = 1, \ldots, m$, are ordered top-to-bottom and, within lines, left-to-right, so that it matches the usual reading flow in documents. Formally, let the top-left corner of a document image refer to the coordinate $(0, 0)$. Further, let us consider the top-left location of all bounding boxes B_j. The top-left location is then used to sort the entities first by their y-coordinate of B_j and, if equal, by their x-coordinate (both ascending). The heuristic o_2 is used for all float environments, including all of their (sub)children.

Appendix D Scalable Weak Supervision

We perform following processing steps to generate noisy labels for weak supervision:

1. Bounding boxes that are retrieved for simple text tokens inside the source code are mapped to CONTENT LINE entities.

2. If we encounter `environments` or `commands` (e.g., `\begin{itemize}` or `\item`), we create corresponding candidate entities. All entities retrieved for tokens inside the scope of these environments are created as nested child entities. This approach is used to create the following entity types, namely FIGURE, FIGURE GRAPHIC, FIGURE CAPTION, TABLE, TABULAR, TABLE CAPTION, ITEMIZE, ITEM, ABSTRACT, and BIBLIOGRAPHY. Any other entities are mapped onto the CONTENT LINE category.

3. We use a set of heuristics that build upon the position, size, alignment and nesting of the returned bounding boxes to generate EQUATION, SECTION, HEADER, CONTENT BLOCK, BIBLIOGRAPHY BLOCK as well as all table structure entities.

4. We utilize a special characteristic of `synctex` to identify EQUATION, EQUATION FORMULA and EQUATION LABEL entities: bounding boxes returned by `synctex` are highly uniform and typically consist of per-line bounding boxes of consistent width and x-coordinates. Equations and labels are an exception to this rule and typically only consist of vertically aligned bounding boxes of smaller width.

5. The sectioning structure of documents is considered: any type of `section` command

---

approach should correctly arrange entities in two cases: (1) in single-column pages and (2) when multi-column pages are split into different chunks by full-width figures or tables.

is mapped to a SECTION entity. The argument of the sectioning command, e. g. `\subsection{titlearg}` is mapped via `synctex` to a HEADER entity. Entities generated from code in the scope of a section are created as children to the section entity that corresponds to the current section scope.

6. Within sections, we sort entities based on a top-to-bottom, left-to-right reading order. Using these sorted lists of sibling entities, we form TEXT BLOCK entities from subsequent groups of CONTENT LINE entities within page columns. If such block occurs within a BIBLIOGRAPHY environment, we instead map it to a BIBLIOGRAPHY BLOCK entity.

7. With an exception of captions, we consider all child entities that do not span across the whole table width as cells and the remainder as table rows. As we shall see later, this is effective at retrieving complex table structures.

8. We use the detected cells to generate rows and columns as follows: We compute the centroids of all cells. To identify rows, we consider the sorted y-coordinates of the centroids and group them such that the pixel-wise distance between two consecutive y-coordinates in a group is smaller or equal to 5. If any identified group contains two or more centroid y-coordinates, we create a TABLE ROW entity from the union of the corresponding table cells. Analogously, using the x-coordinates of the cell centroids, we identify TABLE COLUMN entities.

9. Additional cleaning steps are performed for tables and figures: Child entities with widths or heights of 2 or fewer pixels are discarded. Caption bounding boxes that enclose other non-caption child entities are also discarded.

10. We make sure that entities contain at most one leaf node by moving excess leaves into newly generated CONTENT LINE entities.

11. We remove duplicate bounding boxes and entities without any leaf nodes in their respective sub-tree. Candidates are filtered such that only a group of entities and their respective sub-tree are preserved: itemize, figure, table, equation, heading, content block, bibliography, abstract.

Appendix E   Computational Setup

Training procedure: All neural models are initialized with pre-trained weights from the MS COCO dataset (Lin et al., 2014). We then train each model across three phases for a total of 80,000 iterations. This is split into three phases of 20,000, 40,000, and 20,000 iterations, respectively. During the first phase, we freeze all layers of the CNN that is used as the initial block in Mask R-CNN. In the second phase, stage four and five of the CNN are unfrozen. In the last phase, all network layers are trainable. Early stopping is applied based on the performance on the validation set. The performance is measured every 2000 iterations via the so-called intersection over union (cf. the definition in Equation 1) with a threshold of 0.8.

We train all models in a multi-GPU setting, using 8 GPUs with a vRAM of 12 GB. Each GPU was fed with one image per training iteration. Accordingly, the batch size per training iteration is set to 8. Further, we use stochastic gradient descent with a learning rate of 0.001 and learning momentum of 0.9.

Parameter settings: During training, we sampled randomly 100 entities from the ground truth per document image (i. e., up to 100 entities as some document images might have fewer). In Mask R-CNN, the maximum number of entity predictions per image is set to 200. During prediction, we only keep entities with a confidence score P_j of 0.7 or higher.

Weak supervision: Training with weak supervision is as follows: all models initialized with the weights of our pre-trained DocParser WS instead of default weights. We perform training with learnable parameters analogous to phase 1 above but for 2000 steps with early stopping. In our experiments, we use only a subset of 80 % of the annotated documents from arXivdocs-weak, while the other 20 % remain unused. The intention is that we want to allow for additional annotations in the future while ensuring comparability to our results. We further ensure a fairly uniform distribution of entities by utilizing only document pages that contain at least an abstract, a figure, or table, while all others are discarded. This amounts to 593,583 pages.

Appendix F   Robustness Checks with Table Structure Parsing

We perform robustness checks of DocParser on the table structure parsing task. DocParser is evaluated for both entity detection and structure parsing on arXivdocs-target and the ICDAR 2013 table structure dataset.

F.1 Table Structure Heuristics

For the ordering of table structure entities, we draw upon a set of special heuristics. The reason for this is that nesting relationships are often too complex to model with the previously described parent-child relationships, e. g. for cells belonging to multiple rows and/or columns. Due to these complex relations, bottom-up creation of table row and table column entity bounding boxes from associated children is also challenging. We, therefore, generate rows, columns, and cells on the same hierarchical levels and store structure information in an additional attribute in each entity.

The following heuristics are applied:

1. Rows are sorted, based on the y-coordinate of their centroids. Columns are analogously sorted, based on their centroid x-coordinates.
2. Rows entities that are located such that their bounding box is fully contained inside the bounding box of other row entities are determined. All such direct nestings are resolved as follows: (1) If a row entity contains exactly one other row entity, remove the contained entity. (2) Remove row entities that contain more than one other row entity. Analogously, we proceed to discard column entities with direct nesting.
3. The bounding box (i. e., "union") of all row and column entities is computed. However, the size of this bounding box might differ from the bounding boxes of the row and column entities. Hence, the bounding boxes of all rows are adjusted so that all adjacent rows have the width as the "union". Analogously, the height for all bounding boxes belonging to columns are adjusted.
4. The location of rows might not be located at the center of adjacent rows. This is achieved by setting the y-coordinate of each row to the average of its adjacent rows. An analogous adjustment is performed for the x-coordinates of columns.
5. Row and column numbers are assigned to separately detected cells as follows: for all cell entities from DocParser, we calculate the overlap between the vertical cell border and all vertical row borders. We then calculate the rows for which the length of the overlap is equal or larger than 50 % of the height of a row. The number of the corresponding row is then assigned to the row range of the cell. Analogously, we match cells to columns based on their horizontal overlap. If a cell is matched with more than one row

or column, its bounding box is adjusted such that its borders lie on the grid of row and column borders. All other cells without assignment are dismissed.
6. A grid of rectangular cells is generated from the intersection of all rows and columns for all positions in the table where no multi-row or multi-column cell exists.

F.2 Implementation Details

Entity Detection We use the hierarchical document annotations in arXivdocs-weak to identify 222,195 table structure entities that are used for weak supervision. The corresponding cropped tabular regions and their child entities, i. e., rows, columns, and cells, are used as training input for the specialized system. The sampling process is stratified to bolster prediction performance: we use all row and column annotations, but only a subset of all table cell annotations. The reason is that regular cells can be reconstructed from robust detections of rows and columns. Row and column detection performance can, however, be adversely affected by category imbalance during sampling. The comparably large number of individual table cells per input creates such imbalance. Therefore, we only sample table cells that appear in the first table row and column, as well as cells spanning multiple rows or columns. Altogether, this aids the detection of multi-row and -column cells. Again, these cells can not be robustly reconstructed from regular rows and columns otherwise. The parameters for entity samples per image, ground truth samples per image and maximum number of predictions per image are set to 200, 200 and 400, respectively.

The train, validation and test splits of arXivdocs-target contain 87, 39, and 61 tabular entities, respectively. Crops of the entities are used for training and evaluation of the system specialized for table structure.

ICDAR 2013 Table Structure Dataset: The ICDAR 2013 table structure dataset (Gobel et al., 2013) is designed to evaluate table structure parsing. This dataset is later leveraged as part of our robustness check so that we can evaluate our weak supervision against state-of-the-art approaches for structure parsing. The dataset consists of 123 images, for which structure annotations, including cells, rows, and columns were created. The dataset comes without predefined train/test split; hence,
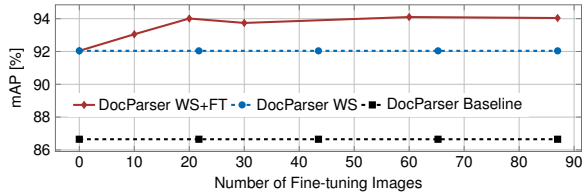
Figure 5: Comparison of test mAP (IoU=0.5) for three variants of DocParser for detection of table structure annotations. We follow the same procedure described in Figure 3. The weakly supervised system Doc-Parser WS outperforms the default system without fine-tuning (FT). Fine-tuning with 10 or more images yields additional performance gains.

we follow Schreiber et al. (2018) and split the so-called "competition" part of the dataset with a $50\%/50\%$-ratio. One of the splits is used for evaluation. The other split is used in addition to the so-called "practice" part of the dataset for training and validation. We follow the official competition rules from ICDAR 2013 as follows: we operate directly on table sub-regions and thus create individual cropped images of these regions for training, validation, and evaluation. We generate rectangular row and column bounding boxes from the provided cell bounding boxes and their respective row- and column ranges. The resulting rows and columns are then further modified as follows: A tabular bounding box is determined as union bounding box of all cells. Bounding boxes of rows that share a border with the outer tabular are extended such that their borders fully align with the tabular. Afterward, we move the borders of all pairs of neighboring rows to their respective midpoint. Analogously, we adjust all column bounding boxes. Cell bounding boxes are newly created from row and column intersections in a final step.

F.2.1 Entity Detection on arXivdocs-target

Analogously to our evaluation on full documents, we measure mAP for table rows and table columns on a subset of table regions in arXivdocs-target. Average precision for joint detection of table rows and columns and the impact of fine-tuning are shown in Figure 5. Compared to full document pages, we measure higher mAPs for all systems. Again, we observe that the weakly supervised model outperforms default DocParser without having been trained on the target domain. We observe additional significant performance improvements in DocParser WS systems that were fine-tuned with 10 to 87 images. Because of the intricacies evaluating hierarchical structure parsing for tables, we

perform a separate evaluation of DocParser for this task.

Appendix-References

Max Gobel, Tamir Hassan, Ermelinda Oro, and Giorgio Orsi. 2013. ICDAR 2013 Table Competition. In *International Conference on Document Analysis and Recognition (ICDAR)*.

Tsung Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision (ECCV)*.

Sebastian Schreiber, Stefan Agne, Ivo Wolf, Andreas Dengel, and Sheraz Ahmed. 2018. Deep-DeSRT: Deep Learning for Detection and Structure Recognition of Tables in Document Images. In *International Conference on Document Analysis and Recognition (ICDAR)*.