BUILDING END-TO-END SYSTEMS FOR HIERARCHICAL DOCUMENT PARSING AND OCR

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES of ETH ZURICH

(Dr. sc. ETH Zurich)

presented by

JOHANNES MARKUS RAUSCH

M.Sc. in Computational Science and Engineering, Technical University of Munich

born on 22.11.1990

citizen of Germany

accepted on the recommendation of

Examiners:

PROF. DR. CE ZHANG (ETH Zurich), examiner
PROF. DR. GUSTAVO ALONSO (ETH Zurich), co-examiner
PROF. DR. STEFAN FEUERRIEGEL (LMU Munich), co-examiner
DR. THEODOROS REKATSINAS (Apple), co-examiner

2023
A dissertation submitted to
ETH Zurich
for the degree of Doctor of Sciences

DISS. ETH NO. 29347

examiner:
Prof. Dr. Ce Zhang

co-examiners:
Prof. Dr. Gustavo Alonso
Prof. Dr. Stefan Feuerriegel
Dr. Theodoros Rekatsinas

Examination date: May 16, 2023
BUILDING END-TO-END SYSTEMS FOR HIERARCHICAL DOCUMENT PARSING AND OCR
ABSTRACT

Information in industry, research, and the public sector is widely stored as rendered documents (e.g. PDF files, scans). Hence, to enable downstream tasks, systems are needed that map rendered documents onto hierarchically-structured text documents. Modern optical character recognition (OCR) systems that are used for this task typically build on two separate stages of parsing the hierarchical document structure and text recognition. Holistic, principled approaches to inferring the complete hierarchical structure of documents were previously missing, due to a range of challenges. A major challenge is given by the complexity of the structure parsing task that consists of the detection of all semantic document entities (e.g. tables, texts, and figures), and relations between entities that describe their hierarchical nesting and reading order. Furthermore, the disjoint setup and use of heuristics in system components hinder training, application, and adaptation in real-world scenarios. Additionally, the annotation of training samples for learning-based systems is very time-consuming and costly.

In this thesis, we work towards solving these issues by building scalable systems that allow unified end-to-end document parsing and optical character recognition. Specifically, this thesis brings the following contributions:

1. DocParser: an end-to-end system for parsing (i) entities in documents (e.g., figures, text blocks, headers) and (ii) relations that capture the sequence and nested structure between entities. Furthermore, we provide a freely accessible dataset for evaluating hierarchical document structure parsing. Finally, we provide a scalable learning framework for settings where domain-specific data are scarce. We address this with a novel approach to weak supervision that significantly improves the document structure parsing performance.

2. Document Structure Generator (DSG): a novel system for document parsing that is fully end-to-end trainable. Previous document structure parsing systems are limited by heuristics and are not end-to-end trainable. The end-to-end training of DSG makes it effective
and flexible for real-world applications. Furthermore, our DSG generates structured document output files in the hOCR markup language, allowing seamless integration into existing document storage and processing workflows. We also contribute a new, large-scale, openly available dataset called E-Periodica comprising real-world magazines with complex document structures for evaluation. Our results demonstrate that our DSG achieves state-of-the-art performance for the hierarchical document parsing task. To the best of our knowledge, our DSG system is the first end-to-end trainable system for hierarchical document parsing.

3. LayTr: a transformer-based system for joint structure parsing and text recognition. Modern state-of-the-art OCR systems still largely rely on separate processing stages for parsing document structure and recognizing texts. This results in several limitations in current end-to-end OCR systems. Individual components must be separately trained and adapted to each other, end-to-end system evaluation is challenging, and structure and language information cannot be used jointly at both stages for effective text recognition and error mitigation. LayTr can be trained fully end-to-end to directly predict marked-up text from document images with complex layouts. We employ evaluations tailored to the end-to-end OCR task and show that our system outperforms state-of-the-art commercial and open-source systems on benchmark datasets.
ZUSAMMENFASSUNG


In dieser Arbeit arbeiten wir an der Lösung dieser Probleme, indem wir skalierbare Systeme entwickeln, die ein einheitliches End-to-End-Parsing von Dokumenten und optische Zeichenerkennung (OCR) ermöglichen. Diese Arbeit liefert insbesondere die folgenden Beiträge:

adressieren dies mit einem neuartigen Ansatz zur schwachen Überwachung, der die Leistung beim Parsen von Dokumentenstrukturen deutlich verbessert.


ACKNOWLEDGMENTS

Above all, I want to express my sincere gratitude to my PhD advisor, Ce Zhang for his invaluable guidance and support throughout my PhD. His dedication and mentorship were crucial to my academic growth and success. I am grateful for your patience and encouragement which helped me navigate the challenges of my doctoral studies. Lastly, thank you for building and shaping such an amazing research lab.

Second, I would like to thank my co-advisor, Gustavo Alonso, for his mentorship and guidance throughout the different stages of my PhD. Thank you for giving me the feeling that I could always reach out for feedback and all of your constructive feedback. I am also very grateful to the members of the examination committee, consisting of Ce Zhang, Gustavo Alonso, Stefan Feuerriegel, and Theo Rekatsinas for their time, effort, and feedback in reviewing my thesis.

I had the privilege to work with other research groups and industry partners during my PhD for internships, research visits, and project collaborations. For this, I’d like to particularly thank Jan Kautz and Thomas Breuel at NVIDIA, Chris Ré and Sen Wu at Stanford, Gero Gunkel at Zurich Insurance, Christiane Sibille at ETH Library, and Stefan Feuerriegel at LMU Munich.

I received great support in my academic career prior to joining ETH which empowered me to do a PhD. I’m very grateful to previous mentors, including Andreas Maier, Rebecca Fahrig, Christian Riess, Waldo Hinshaw, Reza Nezafat, and Björn Menze.

During my PhD, I also had the pleasure to supervise many students for their final theses and research projects. Thank you, particularly to Fabian, Octavio, Gentiana, Maxim, Vennilah, Iveta, Livio, Katerina, Roy, and Jonas.

Thank you to Johannes, Lukas, Melissa, Anastasiia, and David for proofreading different parts of the thesis.

I had the privilege to do my PhD as part of ETH Systems Group and DS3Lab, where I met wonderful co-workers, many of which I became close friends with. I fondly remember the great
times spent with many colleagues, including David, Melissa, Anastasiia, Lukas, Bojan, Jansen, Susie, Maurice, Shaoduo, Andrea, Michal, Cedric, Nora, and Thomas. Thanks a lot as well to our professors Gustavo, Mothy, Ce, Anna, Theo, and Ankit, and our group administrators Nadia, Simonetta, Jena, and Natasha.

Finally, I would like to sincerely express my gratitude to all the individuals that supported and encouraged me throughout my academic journey, whether through their kind words, actions, or simply by being a source of inspiration. I want to deeply thank all of the close people, friends, and family had the privilege to have in my life throughout the years, including Julian, Melissa, David, Johannes, Anastasiia, Ela, Lukas & Lukas, Liesbeth, Andi, Tom, and Chris.

Zurich, 2023
The results presented in this thesis are the product of joint research projects and collaborations, most notably the ones resulting in the following works: (Rausch et al., 2021, 2023a,b). Some descriptions are taken directly from these papers. For this, I’m particularly grateful to the contributions of G. Rashiti, F. Bissig, M. Gusev, O. Martinez, T. Breuel, S. Feuerriegel, and C. Zhang.
CONTENTS

Abstract

Acknowledgments v

1 Introduction 1
   1.1 Contributions .................................................. 4
   1.2 Organization of the Thesis ..................................... 5

2 Background 7
   2.1 Problem description ............................................ 7
      2.1.1 Hierarchical structure parsing ......................... 7
      2.1.2 Document OCR ............................................. 8
      2.1.3 Machine-readable documents ............................ 8
      2.1.4 Denomination ............................................. 9
   2.2 Machine learning background ................................. 9
      2.2.1 Supervised learning: ..................................... 9
      2.2.2 Computer vision background ............................. 10
   2.3 Performance metrics for document structure parsing ....... 12
      2.3.1 Entity detection .......................................... 12
      2.3.2 Hierarchical relation prediction ....................... 12

3 DocParser: Hierarchical Document Structure Parsing from Renderings 13
   3.1 Introduction .................................................. 13
      3.1.1 Contributions ............................................ 15
      3.1.2 Overview ................................................ 16
   3.2 DocParser system .............................................. 16
## Contents

3.2.1 Problem description .................................................. 16
3.2.2 System components .................................................. 16
3.3 Datasets with document structures ................................. 22
  3.3.1 arXivdocs-target .................................................. 23
  3.3.2 arXivdocs-weak .................................................. 23
3.4 Computational setup .................................................. 24
  3.4.1 System variants .................................................. 25
  3.4.2 Performance metrics ............................................. 26
  3.4.3 Robustness check: table structure parsing ..................... 26
3.5 Results ........................................................................ 27
  3.5.1 Document structure parsing .................................... 27
  3.5.2 Robustness check: table structure parsing ..................... 30
3.6 Related work .......................................................... 31
3.7 Discussion .................................................................. 32
3.8 Summary ..................................................................... 33

4 DSG: An End-to-End Document Structure Generator .......... 35
  4.1 Introduction ............................................................ 35
    4.1.1 Contributions ..................................................... 37
    4.1.2 Overview .......................................................... 37
  4.2 Related work .......................................................... 38
  4.3 Problem description .................................................. 39
  4.4 Our DSG system ....................................................... 40
    4.4.1 Image preprocessing (C1) ..................................... 40
    4.4.2 Entity detection (C2) ............................................ 41
    4.4.3 Relation classification and entity refinement (C3) ......... 42
    4.4.4 Grammar-based postprocessing (C4) ....................... 44
    4.4.5 hOCR conversion engine (C5) ............................... 44
    4.4.6 Implementation details ....................................... 46
  4.5 Datasets ................................................................. 48
    4.5.1 arXivdocs-target ................................................. 49
    4.5.2 E-Periodica ....................................................... 49
  4.6 Experimental Setup .................................................... 54
    4.6.1 Performance metrics ............................................. 54
    4.6.2 Training and hyperparameter tuning ......................... 54
4.6.3 Baselines .......................................................... 55
4.7 Results .............................................................. 58
4.7.1 Numerical Experiments ............................................. 58
4.7.2 Qualitative Evaluation ................................................ 62
4.8 Querying .............................................................. 64
4.8.1 hOCR Querying ...................................................... 64
4.8.2 Examples ............................................................. 64
4.9 Discussion ............................................................. 65
4.10 Summary ............................................................... 66

5 LayTr: Differentiable end-to-end OCR for Complex Layouts with Transformers 67
5.1 Introduction ........................................................... 67
5.1.1 Contributions ......................................................... 69
5.1.2 Overview .............................................................. 69
5.2 Related work .......................................................... 70
5.2.1 OCR system components ............................................. 70
5.2.2 Datasets and benchmarks ........................................... 70
5.3 Datasets ................................................................. 72
5.3.1 LayTr-Columns ......................................................... 72
5.3.2 LayTr-Tables .......................................................... 73
5.4 Experimental details ................................................... 75
5.4.1 LayTr system .......................................................... 75
5.4.2 Training procedure .................................................... 77
5.4.3 Compared OCR systems ............................................. 78
5.4.4 Evaluation .............................................................. 79
5.5 Results ................................................................. 80
5.5.1 Quantitative evaluation .............................................. 81
5.5.2 Qualitative evaluation ............................................... 85
5.6 Discussion ............................................................. 85
5.6.1 Performance .......................................................... 85
5.6.2 Datasets ............................................................... 88
5.6.3 Compared systems ................................................... 88
5.6.4 Future work .......................................................... 89
5.7 Summary ............................................................... 89
## Contents

6 Conclusions and Future Work ........................................... 91

A DocParser .......................................................... 93
   A.1 Performance of Document Structure Parsing ................. 93
      A.1.1 Qualitative Evaluation .................................. 93
      A.1.2 Reproducibility ......................................... 95
   A.2 Document Grammar ............................................. 96
   A.3 Datasets with Document Structure: arXivdocs-target ........ 97
   A.4 Datasets with Document Structure: arXivdocs-weak ........ 99
   A.5 Related Work .................................................. 103
      A.5.1 Weak Supervision for Document Layout: .............. 103
   A.6 Robustness Check: Table Structure Parsing ................. 105
      A.6.1 Table Structure Heuristics ............................... 105
      A.6.2 Implementation Details .................................. 107

B Document Structure Generator ...................................... 115
   B.1 DSG System ................................................... 115
      B.1.1 Component 4: Grammar-based postprocessing .......... 115
   B.2 Datasets ....................................................... 116
      B.2.1 E-Periodica ............................................... 116
   B.3 Evaluation ..................................................... 117
      B.3.1 Qualitative evaluation ................................... 117

Bibliography ........................................................... 123
INTRODUCTION

Huge amounts of information are generated daily in industry, research, and the public sector. Yet, such data is typically stored as document renderings (e.g., PDF files, scans) and not as structured hierarchical formats (Johnson, 2018). This is a crucial obstacle for practice. On the one hand, structured formats as opposed to document renderings are needed for efficient storage in databases, as the latter requires standardized formats (Johnson et al., 2003; Clifton et al., 2000). On the other hand, document renderings cannot be processed in downstream tasks since downstream tasks commonly require documents that are in a parsable format. Examples are query and retrieval (Che et al., 2006; Cafarella et al., 2008; Wilkinson, 1994; Li and Moon, 2001; Manabe and Tajima, 2015) and knowledge base construction (Wu et al., 2018).

The structural and layout information in a document can also be a rich source of information that facilitates Natural Language Processing (NLP) tasks (e.g. information extraction). Over the years, the NLP community has developed a range of techniques to detect, understand, and take advantage of document structures (Hurst and Nasukawa, 2000; Chen et al., 2000; Tengli et al., 2004; Luong et al., 2011; Govindaraju et al., 2013; Katti et al., 2018; Schäfer et al., 2011; Zanibbi et al., 2004; Garncark et al., 2021).

A standard solution to map document renderings onto parsable documents are optical character recognition (OCR) systems. For this, the textual contents must be extracted from the document rendering while preserving the semantic and hierarchical structure of the source files. To achieve this, state-of-the-art OCR systems typically operate in two steps: Hierarchical structure parsing and text recognition (see overview in Figure 1.1). Current OCR systems are highly effective in retrieving word-level textual contents from rendered documents (Breuel, 2017). However, OCR systems generally focus on the textual content, and thus struggle with inferring hierarchical
Chapter 1. Introduction

Figure 1.1: Overview of the document OCR process. OCR systems typically rely on a structure parsing step, followed by text recognition to generate hierarchically-structured text documents. Particularly the hierarchical structure parsing step remains challenging.

structure. The step of parsing the document structure is still very challenging and error-prone (Binmakhashen and Mahmoud, 2019). As a consequence, OCR systems suffer from large errors, especially if errors in the step for parsing the structure parsing occur (Zhu et al., 2022). This is due to the complex, nested structures in real-world documents. To this end, there is a direct need in practice (Binmakhashen and Mahmoud, 2019) for systems that map document renderings onto a structured hierarchical format.

The task of document structure parsing refers to the generation of a hierarchical document structure, given a document rendering as input. For this, the textual contents must be extracted from the document rendering while preserving the semantic and hierarchical structure of the source files. To achieve this, state-of-the-art systems typically first detect all document entities (e.g., figures, text blocks, headers) and subsequently infer the hierarchical relationships between entities (e.g., their sequence and nested structure) to form hierarchical document structures. Structure parsing performance is evaluated by an F1 score of classifying hierarchical relations, represented by triples formed by two semantic entities and the relation type. Structure parsing is challenging due to the complex, nested structures in real-world documents.

With respect to structure parsing systems, the majority of existing methods focus only on sub-problems such as the detection of document entities (but not their hierarchical relations) (Zhong et al., 2019) or parsing of partial structures such as table structures (Schreiber et al., 2017). This, however, leads practitioners to rely on disjoint systems that are complicated to use, and hard to
adapt jointly. Particularly, end-to-end systems that can operate on full documents are missing. There exists a great demand for holistic systems for document structure parsing and document OCR to facilitate downstream tasks such as in NLP. The objective of this thesis is to contribute to the research and development of such end-to-end systems.

We address the following three research questions in this thesis:

**Question 1:** How can we build holistic systems for full hierarchical document parsing systems?

Several challenges exist for full hierarchical document parsing systems. First, the task of correctly identifying all the deeply nested structures and document entities is nontrivial due to the complexity of input documents. Second, efficient learning is hindered by the lack of available large-scale training data sets. Third, the annotation of full hierarchical structures is very costly. Lastly, datasets and benchmarks for the evaluation of document parsing are largely missing.

**Question 2:** How can we enable more effective and flexible systems for complex real-world settings?

The use of heuristics in document parsing systems can be a limiting factor in their real-world applicability. Here, systems need to be constantly re-engineered in order to be applicable to documents with yet unseen layouts. Furthermore, datasets that feature more complex layouts are required to advance the development of end-to-end parsing systems. Finally, due to their reliance on heuristics, the structure parsing performance systems can be limited.

**Question 3:** How can we overcome the challenges imposed by the separation of structure parsing and text recognition in OCR?

As stated above, structure parsing by itself brings various challenges and a large need for end-to-end trainable systems exists. In document OCR, hierarchical structure parsing is typically performed separately from text recognition, which poses additional challenges. OCR systems are difficult to engineer and train as separate components need to be tuned and applied jointly. Additionally, textual and structure information is largely treated in an isolated way and cannot be effectively shared across the two OCR sub-components. Lastly, due to the prevailing paradigm of separating structure parsing and text recognition, benchmarks and evaluations that cater to full end-to-end systems are limited.
Chapter 1. Introduction

1.1 Contributions

We provide three major contributions that address the three previously raised main research questions.

(Contribution 1) “DocParser”, a holistic system for full hierarchical document parsing. Our system builds on a deep learning architecture to tackle the demanding task of identifying the deeply nested structures in documents. We contribute novel large-scale datasets: arXivdocs-weak and arXivdocs-target; for the training and evaluation of full hierarchical document parsing systems. To reduce the high labeling complexity, we leverage a weak supervision mechanism that reduces the labeling complexity by a factor of 8 compared to the baseline systems.

(Contribution 2) “Document Structure Generator” (DSG), a fully end-to-end trainable system for hierarchical document parsing. DSG overcomes restrictions of previous systems by replacing the heuristic-based structure recognition component with a machine learning model. DSG interconnects document entity detection and structure recognition in an end-to-end trainable system that does not require re-engineering when applied to new datasets. We contribute the new large-scale E-Periodica dataset of complex, real-world scanned magazines to further development and evaluation of systems. Finally, we demonstrate the effectiveness of the DSG system, achieving state-of-the-art performance for hierarchical structure parsing.

(Contribution 3) “LayTr”, an end-to-end document OCR system for joint structure parsing and text recognition. LayTr does not rely on heuristics or individual components for structure parsing recognition and text recognition. Instead, it directly produces marked-up text as output by taking in document renderings. Furthermore, our system design enables it to jointly leverage structural and textual content for the end-to-end OCR task. We perform evaluations for end-to-end document OCR and show that LayTr outperforms state-of-the-art commercial and open-source OCR systems, as measured by end-to-end string edit distance metrics, and presents a promising approach for unified hierarchical document parsing and text recognition.

Publications

This dissertation is largely based on the following publications presented in the order of appearance of this thesis:
1.2 Organization of the Thesis


1.2 Organization of the Thesis

This thesis is organized as follows: We provide a shared problem description and background in Chapter 2. Chapter 3 introduces DocParser, a system for full hierarchical document parsing that addresses the lack of existing, effective end-to-end systems, datasets, and high labeling complexity. Chapter 4 describes DSG, a document parsing system that is fully end-to-end trainable and the new E-Periodica dataset. In Chapter 5 the system LayTr is presented. LayTr builds on a transformer-based model that allows for unified document layout analysis and text recognition. We show that LayTr can be trained for effective parsing of complex document inputs. Chapter 6 features a conclusion and summary of the thesis.
BACKGROUND

In this chapter, we introduce the background shared across multiple main chapters. Chapters 3, 4 and 5 all relate to the general task of document structure parsing. We introduce a problem description that is applicable to all of the chapters. We slightly extend the description in some Chapters to account for differences in system design. Chapter 5 features a system to tackle the task of end-to-end document OCR, which commonly involves document structure parsing, as well as text recognition.

2.1 Problem description

Document OCR and hierarchical structure parsing describe two related tasks. Current state-of-the-art document OCR systems typically incorporate a hierarchical structure parsing, as well as a text recognition component. This section aims to provide problem descriptions to differentiate the individual tasks. Furthermore, denominations for individual tasks can vary in the literature. For this reason, we use a consistent denomination to be used throughout the thesis.

2.1.1 Hierarchical structure parsing

Objective: The objective of document structure parsing systems, including the ones featured in chapters 3 and 4, is to generate hierarchical document structures from document renderings (e.g., PDF files, scanned images). Formally, the input is given by document renderings \( D_1, \ldots, D_n \) and the outputs are given by hierarchical structures \( H_i, i = 1 \ldots, n \). The hierarchical
structure is defined by a set of (i) entities in the documents (e.g., figures, text blocks, headers, etc.) and (ii) relations that capture the sequence and nested structure between entities. Formally, entities are given by \( E_j, j = 1, \ldots, m \), and relations by \( R_j, j = 1, \ldots, k \). Both are defined below.

**Entities:** Entities capture the different structural elements in documents, such as figures, tables, captions, text blocks, etc. Each entity \( E_j, j = 1, \ldots, m \) is described by four attributes: (1) a semantic category \( c_j \in C = \{ C_1, \ldots, C_l \} \) (e.g., whether it is a figure, table, header, etc.); (2) a rectangular bounding box \( B_j \) in the document rendering, defined by the \( x \)- and \( y \)-coordinates of corner points of the bounding box; and (3) a confidence score \( P_j \) that accompanies the prediction of the semantic category \( c_j \).

**Relations:** Relations capture the nested structure among the entities. Relations \( R_j, j = 1, \ldots, k \) of type \( \Psi \) are defined by triples \( (E_{\text{subj}}, E_{\text{obj}}, \Psi) \) consisting of a subject \( E_{\text{subj}} \), an object \( E_{\text{obj}} \), and a relation type \( \Psi \in \{ \text{parent of}, \text{followed by}, \text{null} \} \). Furthermore, the relations \( R_j \) have a confidence score \( P_j^{\Psi} \) for the predicted relation type \( \Psi \).

### 2.1.2 Document OCR

**Objective:** The objective of document OCR is to output hierarchically-structured, marked-up text documents \( M_1, \ldots, M_n \), given document renderings \( D_1, \ldots, D_n \). Marked-up text documents \( M_i \) are formed from pairs \( (H_i, T_i), \ldots, (H_n, T_n) \) (as described above), where \( H_i, i = 1, \ldots, n \) captures the hierarchical structure and \( T_i, i = 1, \ldots, n \) the document texts. Novel systems for end-to-end OCR, such as the one presented in Chapter 5, directly produce marked-up text documents \( M_j \) from document renderings \( D_j \) without building intermediate representations \( (H_j, T_j) \).

**Document texts:** Document texts \( T_i, i = 1, \ldots, n \) contain the textual contents of the document renderings \( D_i \). Document texts are defined by a set of text entities in the document. Formally, the text entities are given by \( W_j, j = 1, \ldots, q \). Each text entity \( W_j, j = 1, \ldots, q \) is described by two attributes: (1) the written character or text represented by the entity, typically using Unicode or ASCII encoding. (2) An optional rectangular bounding box \( B_j \) in the document rendering, defined by the \( x \)- and \( y \)-coordinates of corner points of the bounding box. Text recognition algorithms typically work on a per-word or per-line basis.

### 2.1.3 Machine-readable documents

The systems presented in this thesis operate on document rendering inputs. For completeness, we additionally describe the setting encountered for machine-readable input files \( D_1^{\text{MR}}, \ldots, D_n^{\text{MR}} \),
2.2. Machine learning background

such as born-digital PDF files. For these input files, typically no text recognition step is required and document texts $T_i$ can be directly retrieved. Furthermore, such documents can already be accompanied with additional logical-structure information (e.g. tagged PDF (ISO/TC 171, 2014)). However, if no such structure is given, hierarchical structure parsing is used to retrieve hierarchical structures $H_i$. Here, analogously to document OCR, the objective is to form pairs $(H_1, T_1), \ldots, (H_n, T_n)$ or marked-up text documents $M_1, \ldots, M_n$ from machine-readable documents $D^\text{MR}_1, \ldots, D^\text{MR}_n$.

2.1.4 Denomination

Due to the wide range of research fields that feature document parsing and OCR tasks, the denomination of different subtasks can vary. For instance, "OCR" can describe the recognition of document texts $T_i$, the retrieval of the pairs $(H_i, T_i)$, or the marked-up texts $M_i$. In the context of OCR research, the terms "segmentation" or "layout analysis" are often used to describe either the detection of entities $E_j$ or full hierarchical structures $T_i$. Furthermore, "segmentation" can be used to describe entities by bounding boxes $B_j$, or by their pixel-wise coordinates on the document rendering. For this reason, we use the consistent denomination as described above in the following chapters.

2.2 Machine learning background

Machine learning techniques can be used to automate the process of hierarchical document structure parsing. Here, a typical setting is the training of a machine learning system from manually annotated hierarchical document structures. After the training process, the machine learning system can then be used to parse new input documents and automatically output document structures.

2.2.1 Supervised learning:

The objective of supervised learning methods is to make predictions on unseen data, given labeled data. Given a feature space $X$ and output space $Y$, and $n$ data points $(x_i, y_i), i = 1\ldots,n$ with $x_i \in X, y_i \in Y$, we aim to learn a mapping function $g : X \rightarrow Y$ from feature to output space. We refer to the set of $n$ data points as a dataset $D := (x_i, y_i)_{i=1}^n$. Input features $x_i$ are typically represented as feature vectors of dimension $d$ with $x \in \mathbb{R}^d$. The outputs $y_i$ represent the label
Chapter 2. Background

associated with the respective features \(x_i\). Depending on the task at hand, the outputs can, for instance, be categorical labels with \(Y = 1, \ldots, C\) for classification tasks with a total of \(C\) classes. Continuous labels \(Y = \mathbb{R}\) are used for regression tasks. It is assumed that the training dataset \(D\) consists of independent and identically distributed random data points \((x_i, y_i)\).

The experiments conducted in this thesis mainly feature supervised learning algorithms. We also consider weak supervision, where data points might have partially inaccurate labels. This means that not all labels in the training dataset \(D\) might be incorrect. This can be useful in settings where machine learning models might still benefit from larger-scale datasets of noisy data, in the absence of large-scale datasets for fully supervised learning.

A loss function \(L : Y \times Y \to \mathbb{R}_0^+\) is used to measure the loss, i.e., the loss \(L(y_i, \hat{y}_i)\) that results from predicting a value \(\hat{y}_i\), given a data point with the label \(y_i\). The concrete choice of loss function depends on the specific type of machine learning task at hand. For instance, a cross-entropy loss \(L_{CE} = (y_i \log(\hat{y}_i)) + (1 - y_i) \log(1 - \hat{y}_i)\) can be used for classification problems. For regression problems, commonly used loss types are the L1 loss \(L_{L1} = |\hat{y}_i - y_i|\), or L2 loss \(L_{L2} = (\hat{y}_i - y_i)^2\).

It is common practice in machine learning experiments to split datasets into three separate parts, called training, validation (or development), and test sets. During training, a machine learning model is only exposed to data points that belong to the training set. As part of the experimental procedure, a validation set is used to evaluate the model and to optimize model hyperparameters such as the learning rate. This is done to avoid overfitting on the test set.

2.2.2 Computer vision background

Advancements in neural network architectures (Krizhevsky et al., 2012; Lecun et al., 1998; Deng et al., 2009; He et al., 2016; Girshick, 2015), large-scale datasets (Deng et al., 2009; Kuznetsova et al., 2020; Krishna et al., 2017; Lin et al., 2014), as well as computational resources and parallelization (Krizhevsky et al., 2012; Dosovitskiy et al., 2023) have led to significant advancements in the field of computer vision.

Models that are used in computer vision typically build on deep neural networks, specifically convolutional neural networks (CNN) (Krizhevsky et al., 2012; Simard et al., 2003). More recently, transformer-based architectures have been introduced (Dosovitskiy et al., 2023; Vaswani et al., 2017).
An important task in computer vision is entity detection\(^1\). The objective is to localize entities in an input image, whereas entities \(E_j\) are specified by their rectangular bounding box \(B_j\) and category \(c_j\).

Here, a common architecture type for entity detection is Faster R-CNN (Ren et al., 2015). We provide a brief overview of this architecture to introduce the reader to the models that are featured in chapters 3 and 4 of this thesis.

![Faster R-CNN architecture](image)

**Figure 2.1:** Faster R-CNN architecture.

Faster R-CNN architectures commonly extend the architecture of a convolution neural network (He et al., 2016) so that they are highly effective for image and entity detection. Formally, it comprises multiple stages with decreasing spatial resolution. The output of these stages is then fed into a so-called feature pyramid network (FPN) (Lin et al., 2017). The FPN then interconnects these inputs in multiple stages of increasing spatial resolution to produce multi-scale feature maps. The underlying convolutional neural network architecture (He et al., 2016) extracts features in 5 stages at different resolutions. The outputs of stages 2 to 5, denoted as \(C_2, \ldots, C_5\), are passed to the FPN. The FPN outputs a total of 5 feature maps \(P_2, \ldots, P_6\) at different resolutions. We refer the reader to (Lin et al., 2017) for a detailed description of the five feature maps. The multi-scale feature maps are then input to different prediction networks: first, a region proposal network (RPN) generates a list of candidate bounding boxes that should contain an entity. Second, a Region of Interest (RoI) alignment layer filters out the multi-scale feature maps that correspond to the candidate regions. We note that all 5 feature maps are used by the RPN, but \(P_6\) is not included in the inputs to the RoI alignment layer. Third, these bounding boxes are subsequently refined in a detection sub-network, thereby yielding the final bounding boxes \(B\) and categories. It also provides the label for the entity category.

---

\(^1\)For consistency in terminology, we use the term entity detection throughout the thesis. In the computer vision domain, this task is commonly referred to as object detection.
2.3 Performance metrics for document structure parsing

Chapters 3 and 4 present systems for document structure parsing. We describe two common performance metrics for such systems as follows.

2.3.1 Entity detection

Entity detection is commonly measured by the mean average precision (mAP) of a model (0: worst, 100: best). The inferred entities $E_j = (c_j, B_j, P_j)$ are 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., 2009) and measure the overlap between bounding boxes from the same category. Specifically, we calculate the so-called intersection-over-union (IoU) via

$$\text{IoU} = \frac{\text{area}(B_j \cap \hat{B}_j)}{\text{area}(B_j \cup \hat{B}_j)}. \quad (2.1)$$

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 entity 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 $C_k \in C$. The overall performance across all categories is given by the mean average precision.

2.3.2 Hierarchical relation prediction

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 only if the complete tuple is identical. The performance depends on the correct entity detection as input. Hence, IoU thresholds for entity detection are adjusted analogous to the entity detection and then the corresponding F1 score for correctly predicting hierarchical relations is reported. The F1 score is the harmonic average of precision and recall for predicting these triples (0: worst, 1: best).
This chapter is based on the paper (Rausch et al., 2021), which is a joint work with Octavio Martinez, Fabian Bissig, Ce Zhang and Stefan Feuerriegel. Fabian Bissig and Octavio Martinez contributed to dataset and experiments as part of their theses/projects under my supervision. To better contextualize the systems in Chapters 3 and 4, their related work is left intact.

3.1 Introduction

As highlighted in Chapter 1, the structural information in documents can be rich source of information for NLP tasks, and facilitate tasks such as query and retrieval.

However, structural information in documents is becoming increasingly challenging to obtain — many file formats that are prevalent today are being rendered without structural information. Prominent examples are PDF documents: this file format benefits from portability and immutability, yet it often 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 NLP. In contrast, file formats such as XML or JSON naturally encode hierarchical document structures among textual entities. Hence, techniques are required in order to convert renderings into structured, textual document representations to enable joint inference between text, layout, and other document structures.
Earlier attempts for structure parsing on documents focused on a subset of simpler tasks such as segmentation of text regions (Antonacopoulos et al., 2009a), locating tables (Zanibbi et al., 2004; Embley et al., 2006), or parsing them (Schreiber et al., 2017), but not parsing complete document structures.

However, document structures are required as a representation of many downstream tasks in NLP. For instance, recent efforts in the NLP community (Katti et al., 2018; Apostolova and Tomuro, 2014; Liu et al., 2019a) have shown that utilizing 2D document information, e.g. character and word positions, can be an effective way to improve upon standard NLP tasks such as information extraction.

A holistic, principled approach for inferring the complete hierarchical structure from documents is missing. On the one hand, such a task is nontrivial 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., 2017). On the other hand, efficient learning is prevented as large-scale training sets are lacking (cf. Arif and Shafait, 2019; Schreiber et al., 2017). 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). Therefore, an effective implementation that makes only a scarce use of labeled data is demanded.

The work presented in this chapter focuses on parsing the hierarchical document structure from renderings. We develop an end-to-end system for inferring the complete document structure (see...
3.1. Introduction

Figure 3.1). This includes all entities (e.g., text, bibliography regions, figures, equations, headings, tables, and table 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 the 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.

3.1.1 Contributions

We extend prior literature on document parsing in the following directions\footnote{Source codes and the arXivdocs dataset are available from https://github.com/DS3Lab/DocParser.}:

1. We contribute “DocParser”. This presents the first end-to-end system for parsing renderings into hierarchical document structures. Prior literature has merely focused on simpler tasks such as table detection or table parsing but not on the parsing of complete documents. As a remedy, we present a system for inferring document structures in a holistic, principled manner.

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 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 \TeX source files from arXiv together with \texttt{synctex} for this objective. This then yields weakly-supervised labels by reverse rendering of the \TeX code.
4. We conduct extensive evaluation of our proposed techniques, outperforming the state-of-the-art on the related task of table parsing.

3.1.2 Overview

We provide an overview of the DocParser system in Section 3.2. Section 3.3 describes our two contributed datasets, including a range of dataset statistics. We provide details on the computational setup of our experiments in Section 3.4 and present our results in Section 3.5. Relevant related work is discussed in Section 3.6. Finally, we discuss our findings and provide a summary of this chapter in Section 3.7 and Section 3.8, respectively.

3.2 DocParser system

3.2.1 Problem description

We refer to Section 2.1 for a general problem description for hierarchical structure parsing. Given a set of document renderings $D_1, \ldots, D_n$, the objective is to generate hierarchical structures $H_1, \ldots, H_n$. A hierarchical structure $T_i$, $i = 1, \ldots, n$, consists of both entities and relations. Our system differs from the general problem description in that the relations $R_j$, $j = 1, \ldots, k$ of type $\Psi \in \{\text{parent of}, \text{followed by}, \text{null}\}$ do not have an additional confidence score.

3.2.2 System components

DocParser performs document structure parsing via 5 components (see overview in Figure 3.2): (1) image conversion, (2) entity detection, (3) relation classification, (4) structure-based refinement, and (5) scalable weak supervision. To store document structures, we developed a customized, JSON-based file format.

Component 1: Image conversion

Document renderings are converted into images with a predefined resolution $\rho$. Furthermore, all images are resized to a fixed rectangular size $\phi$ (if necessary, with zero padding).

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

Component 2: Entity detection

To detect all document entities within a document image, we build upon a neural model for image segmentation, namely Mask R-CNN (He et al., 2017). Specifically, it takes the images from the previous component as input and then returns a flat list of entities $E_1, \ldots, E_m$ as output. For each entity Mask R-CNN determines (i) its rectangular bounding box, (ii) confidence score, (iii) a binary segmentation mask that distinguishes between the detected entity and background pixels within the bounding box, and (iv) a category label for the entity. Our implementation makes use of 23 categories $C$: CONTENT BLOCK, TABLE, TABLE ROW, TABLE COLUMN, TABLE CELL, TABLEULAR, FIGURE, HEADING, ABSTRACT, EQUATION, ITEMIZE, ITEM, BIBLIOGRAPHY BLOCK, TABLE CAPTION, FIGURE GRAPHIC, FIGURE CAPTION, HEADER, FOOTER, PAGE NUMBER, DATE, KEYWORDS, AUTHOR, AFFILIA-

Component 3: Relation classification

A set of heuristics is applied to translate the flat list of entities into hierarchical relations $R_1, \ldots, R_k$. Here, we distinguish the heuristics according to whether they generate (1) the nesting among

\footnote{For consistency, this formatting is utilized for all entities.}
entities or (2) the ordering for entities of the same nesting level. The former case corresponds to $\Psi = \text{parent} \ of$, while the latter determines all relations with $\Psi = \text{followed} \ by$. In this component, we ignore all entities with meta-information, e.g. footers, as these have no designated hierarchy (cf. document grammar in the Appendix).

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

**($h_1$: Overlaps)** A list of candidate parent-child relations is compiled based on the overlap of bounding boxes. That is, DocParser loops over all bounding boxes and, for each bounding box $B_{\text{subj}}$, it determines all other bounding boxes that are contained within $B_{\text{subj}}$.

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 the bounding box of $B_{\text{obj}}$ is fully or partially enclosed by the bounding box of $B_{\text{subj}}$ are added to the candidate list. Furthermore, we add tuples to the candidate list that satisfy $\frac{\text{area}(B_{\text{subj}} \setminus B_{\text{obj}})}{\text{area}(B_{\text{subj}})} \geq \theta_1$ and $\frac{\text{area}(B_{\text{subj}})}{\text{area}(B_{\text{obj}})} > \theta_2$, i.e. they must have a certain overlap fraction $\theta_1$ and size ratio $\theta_2$. 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 a predefined document grammar (see document grammar in the Appendix). Concretely, all illegal candidates, e.g., a tabular nested inside a figure, 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 $E_{\text{subj}}$.

**($h_4$: Unique parents)** The candidate list is altered so that each entity has only a single parent. Formally, if an entity $E_{\text{obj}}$ has multiple candidate parents, we first compare the Intersection over Union (IoU) of the bounding boxes of all candidate parents with $E_{\text{obj}}$: $\text{IoU} = \frac{\text{area}(B_{\text{subj}} \cap B_{\text{obj}})}{\text{area}(B_{\text{subj}} \cup B_{\text{obj}})}$ (cf. Equation 2.1). 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 (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. By default, all entities are processed by both heuristics. Children of floating entities are only processed by heuristic $o_2$, however.
(o₁: Page layout entities) First, all entities are grouped according to their coordinates on the document page, 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., \( \tau_{\text{ovlp}} = \text{overlap}/\text{width}(B) \). If the overlap with either the left (or the right) side is above a 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 essence, the center group is an indicator whether the document is in single- or multi-column.

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 \( G_{c,l}, G_{c,c}, \) and \( G_{c,r} \), respectively. Afterward, we loop over all vertical subgroups \( \iota \). For each, we order the entities according to the group (first \( G_{\iota,l} \), followed by \( G_{\iota,c} \) and then \( G_{\iota,r} \)). Within each subgroup, we perform the ordering via heuristic \( o_2 \). This 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.

For each subgroup, we perform the ordering via heuristic \( o_2 \).

(o₂: Reading flow) 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 \((0, 0)\). Furthermore, 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).

Component 4: Structure-based refinement We utilize the classified relations to iteratively refine entities and relations in four steps when parsing full document pages:

(1) For each entity \( E_{\text{parent}} \) with \( l \) child entities \( E_{\text{child}}^1, \ldots, E_{\text{child}}^l \), we update its bounding box such that \( B_{\text{parent}} = \text{union}(B_{\text{parent}}, B_{\text{child}}^1, \ldots, B_{\text{child}}^l) \). (2) For parent entities \( E_{\text{parent}} \) with exactly one child entity of the same category, we remove the child entity and update \( B_{\text{parent}} \) such that it is the union of parent and child bounding boxes. We also consider entity pairs of categories that do not conform to the document grammar. This allows us to dismiss duplicate entities of any
category. (3) If an entity $E_{\text{child}}$ is sibling to other entities in a way that conflicts the document grammar, we generate a new entity that encloses $E_{\text{child}}$ to achieve conformity with the document grammar. Concretely, nested \texttt{figure} structures are defined such that one \texttt{figure} should at most contain one \texttt{figure} \texttt{graphic} entity child. If multiple \texttt{figure} \texttt{graphic} are classified as children, we wrap each of them individually into new \texttt{figure} entities. (4) If no parent is found for an entity $E_{\text{child}}$ that should only occur as a child entity, we identify a suitable parent entity by analyzing its neighboring siblings as follows: we consider all entities that jointly appear in an ordering relation with $E_{\text{child}}$ as a candidates $E_{\text{cand}}$. We dismiss candidates of category $C$ that would not conform to the hierarchies defined in the document grammar. Finally, we dismiss any candidate for which $B_{\text{cand}} \cap B_{\text{child}} = \emptyset$. If exactly one candidate remains, we update its bounding box $B_{\text{cand}} = \text{union}(B_{\text{cand}}, B_{\text{child}})$.

The updates to the set of entities can lead to further changes to the classified relations. For this reason, whenever changes are made to entities in one of the four refinement steps, we update the relations via Component 3 and move back to refinement step (1). The refinement is completed once no change is applied in any of the steps or a limit of $r$ loop iterations has been reached.$^3$

\textbf{Component 5: Scalable weak supervision}

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

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 process has three particular characteristics: first, the mapping is noisy and thus creates only weak labels. Despite that, the weak labels can 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. Furthermore, weak labels are generated merely for a subset $C' \subseteq C$ of the semantic categories.

In DocParser, the weak supervision is based on \TeX source files that are used to generate document renderings in the form of PDF files. The mapping between both formats is then obtained

$^3$Details on our parameter choice and pseudocode are included in the Appendix.
via synctex (Laurens, 2008). synctex is a synchronization tool that performs a reverse rendering, so that PDF locations are mapped to \TeX code. For given coordinates in 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 the 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 \TeX commands. We then map the source code to our entities \( E \). For instance, the bounding box for \TeX code \texttt{\includegraphics} inside a \texttt{\begin{figure} \end{figure}} environment is mapped onto a \texttt{figure} graphic entity that is nested inside a \texttt{figure} entity. Bounding boxes for all entities that act as inner children are created dynamically by computing the union bounding of all child bounding boxes.

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 \texttt{content line} entities.

2. If we encounter \texttt{environments} or \texttt{commands} (e.g., \texttt{\begin{itemize}} or \texttt{\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 \texttt{figure}, \texttt{figure graphic}, \texttt{figure caption}, \texttt{table}, \texttt{tabular}, \texttt{table caption}, \texttt{itemize}, \texttt{item}, \texttt{abstract}, and \texttt{bibliography}. Any other entities are mapped onto the \texttt{content line} category.

3. We utilize a special characteristic of synctex to identify \texttt{equation}, \texttt{equation formula} and \texttt{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.

4. The sectioning structure of documents is considered: any type of \texttt{section} command is mapped to a \texttt{section} entity. The argument of the sectioning command, e.g. \texttt{\subsection{titlearg}} is mapped via synctex to a \texttt{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.

5. 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 \texttt{content 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.

6. In table environments, we consider all child entities (except captions) that do not span across a whole table width as cell and the remainder as table row. As we shall see later, this is effective at retrieving complex table structures.

7. We use the detected table 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 cell entities. Analogously, using the x-coordinates of the cell centroids, we identify table column entities.

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

9. We make sure that entities contain at most one leaf node by moving excess leaves into newly generated content line entities.

10. 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.

During training, entities with obvious errors are dismissed, i.e. leaf nodes or entities with bounding boxes that extend beyond page limits or with area of 0.

### 3.3 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.
3.3. Datasets with document structures

3.3.1 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 362 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. 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. On average, each document contains 86.32 entities. The number of leaf nodes in the document graph as well as the frequency and average depth of the different entities are reported in the Appendix.

Evidently, the most common category in the dataset is content line (34.33%). This is because they typically represent leaf nodes in the graph and are children of larger entities such as abstract, caption, or content block.

Annotators were instructed to follow the document grammar during labeling. Annotation of disallowed hierarchies is, however, possible to provide them the freedom to deal with the range of different document representations. Document annotations are automatically initialized by our scalable weak supervision mechanism to speed up the annotation process. The labelers were instructed to annotate entities only up to the coarseness that is used by DocParser, e.g. labeling content blocks, rather than individual lines.

3.3.2 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.

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

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.
3.4 Computational setup

Our system uses Mask R-CNN, which is a modification of the general Faster R-CNN architecture presented in 2.2.2. We present the modified architecture with its most relevant adaptations as follows.

Our used Mask R-CNN architecture is based on the Faster architecture that is described in Section 2.2.2 (see Figure 2.1 for an overview). Mask R-CNN extends the architecture of a convolution neural network with skip connections (He et al., 2016) so that it is highly effective for image segmentation and entity detection. Specifically, we use a ResNet-110 architecture (He et al., 2016) to extract features in 5 stages at different resolutions with the feature pyramid network. In addition to the bounding box regression and class prediction, our architecture uses the outputs of the RoI alignment layer as input for a mask sub-network. For each region proposal, the mask sub-network predicts the segmentation masks, based on the RoI aligned features. These segmentation masks are not used in subsequent steps of DocParser at prediction time; however, they are utilized in our loss function during the training process.

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. The reason for this modification is that document entities (as opposed to classical image segmentation) contain entities that have highly rectangular shapes. This is the case for most entities, e.g., single content line or table row entities. (2) The output size of the classifier sub-network is modified so that it can produce predictions for entities across all semantic categories \( C \). (3) During training of the mask sub-network, we treat all pixels in ground truth bounding boxes as foreground. We do this to incorporate our understanding of the exact shape of many entities that span very wide rectangular regions. (4) We use a mask sub-network loss with a weighting factor of 0.5. This is to prioritize that features relevant for the correct prediction of bounding boxes and entity categories are learned. The Mask R-CNN stage of DocParser comprises 63,891,032 parameters and is built upon the implementation of Mask R-CNN provided by (Abdulla, 2017), yet which we carefully adapted as described above.

Training procedure: All neural models are initialized with pre-trained weights based on 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, stages four and five of the CNN are unfrozen. In the last
3.4. Computational setup

phase, all network layers are trainable. Early stopping is applied based on the performance on
the validation set for unrefined predictions. The performance is measured every 2000 iterations
via the so-called intersection over union 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. Furthermore, 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 are initialized
with the weights of our pre-trained DocParser WS instead of default weights. We perform the
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 80cedure 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

3.4.1 System variants

We compare the following variants of DocParser: DocParser Baseline is trained solely on
the noise-free labels provided for the training dataset (here: arXivdocs-target); 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; DocParser WS+FT is initialized with the weights from DocParser WS, but then
fine-tuned (FT) on the target dataset.
3.4.2 Performance metrics

We separately evaluate the performance of our system for (i) detection of entities $E_j$ and (ii) classification of hierarchical relations $R_j$. The former aims at a high detection rate (i.e. recognizing true positives out of all positives). Hence, we use the average precision as evaluation metric. The latter is based on the F1 score as it represents a typical classification task (i.e. recognizing one of the relations from $\Psi$).

**Entity Detection:** We measure the accuracy of our entity detection with the average precision (AP) mean average precision (mAP), as described in 2.3.1. We compare IoU thresholds of 0.5 and 0.65.\(^6\)

**Prediction of hierarchical relations:**

For the evaluation of our hierarchical relation prediction, we use the F1 metric described in 2.3.2. Note that our performance measure is relatively strict. We later vary the IoU thresholds for entity detections analogous to above. We show that, even if some F1 scores are in a lower range, we can recover the overall document structure successfully. In particular, we outperform state-of-the-art OCR results, as illustrated in the qualitative samples in our Appendix.

3.4.3 Robustness check: table structure parsing

We additionally train our model for structure parsing so that it identifies table structures to demonstrate the robustness of our system and weak supervision.

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 with the state-of-the-art. The ICDAR 2013 dataset consists of a variety of real-world documents and is not limited to scientific articles. We proceed analogously to full document structure parsing and train the three system variants for the task of table structure recognition.

**DocParser Baseline** is trained solely on the samples provided in the ICDAR 2013 training dataset; **DocParser WS** is trained on table structures generated from arXivdocs-weak. **DocParser WS+FT** is generated by subsequent fine-tuning on the ICDAR training split.\(^7\)

---

\(^6\)Additional results for IoU=0.8 are in the Appendix.

\(^7\)Details about the setting and additional experiments are provided in the Appendix.
Both training and fine-tuning of all variants follow the 3 phase training scheme for a total of 80,000 iterations.\textsuperscript{8}

3.5 Results

The key focus of our experiments is to confirm the effectiveness of DocParser for parsing the complete 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.

3.5.1 Document structure parsing

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

3.5.1.1 Entity detection

The overall performance for entity detection is detailed in Table 3.1 (first row). We discuss the performance for IoU = 0.5 in the following. DocParser Baseline achieves an mAP of 49.9. This is higher than DocParser WS with an mAP of 34.6. 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 69.4, which, in comparison to the baseline DocParser, is an improvement by 39.1 %.

DocParser WS+FT consistently outperforms the baseline system, even for categories that are not annotated during weak supervision (e.g. AUTHOR, FOOTER, HEADER, PAGE NUMBER). We attribute this to the better model initialization due to the prior weakly supervised pre-training.

\textsuperscript{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.
There is a small number of entity categories for which the Baseline achieves higher AP values. We attribute this to our experimental protocol which yields the best model via early stopping, based on mAP and not on individual entity AP values. For a few entities a decrease can be observed after fine-tuning (e.g. `table` at IoU=0.5). We attribute this to the high quality of weak annotations for this category and, consequently, a slight decrease of generalization due to fine-tuning. Some AP values (for both `DocParser` Baseline and `DocParser WS`) amount to 0.0, e.g. for `date`. This is caused by the absence of some categories in arXivdocs-weak in the case of

<table>
<thead>
<tr>
<th>Entity</th>
<th>IoU=0.5 Baseline</th>
<th>IoU=0.5 WS</th>
<th>IoU=0.5 WS+FT</th>
<th>IoU=0.65 Baseline</th>
<th>IoU=0.65 WS</th>
<th>IoU=0.65 WS+FT</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean AP</td>
<td>49.9</td>
<td>34.6</td>
<td><strong>69.4</strong></td>
<td>38.5</td>
<td>32.4</td>
<td><strong>56.5</strong></td>
</tr>
<tr>
<td>ABSTRACT</td>
<td><strong>95.2</strong></td>
<td>90.5</td>
<td>95.2</td>
<td>90.5</td>
<td>81.0</td>
<td><strong>95.2</strong></td>
</tr>
<tr>
<td>AFFILIATION</td>
<td><strong>51.6</strong></td>
<td>0.0</td>
<td>46.0</td>
<td>5.9</td>
<td>0.0</td>
<td><strong>16.2</strong></td>
</tr>
<tr>
<td>AUTHOR</td>
<td>18.0</td>
<td>0.0</td>
<td><strong>23.6</strong></td>
<td><strong>20.4</strong></td>
<td>0.0</td>
<td>16.7</td>
</tr>
<tr>
<td>BIB. BLOCK</td>
<td>42.4</td>
<td>79.1</td>
<td><strong>94.7</strong></td>
<td>43.2</td>
<td><strong>93.9</strong></td>
<td>80.3</td>
</tr>
<tr>
<td>CONT. BLOCK</td>
<td><strong>89.3</strong></td>
<td>69.8</td>
<td>88.4</td>
<td>83.2</td>
<td>67.0</td>
<td><strong>84.4</strong></td>
</tr>
<tr>
<td>DATE</td>
<td>0.0</td>
<td>0.0</td>
<td><strong>24.1</strong></td>
<td>0.0</td>
<td>0.0</td>
<td><strong>9.3</strong></td>
</tr>
<tr>
<td>EQUATION</td>
<td>65.8</td>
<td>54.5</td>
<td><strong>82.1</strong></td>
<td>40.6</td>
<td>52.1</td>
<td><strong>72.8</strong></td>
</tr>
<tr>
<td>FIG. CAPTION</td>
<td>47.8</td>
<td>30.5</td>
<td><strong>69.2</strong></td>
<td>44.0</td>
<td>17.7</td>
<td><strong>59.5</strong></td>
</tr>
<tr>
<td>FIG. GRAPHIC</td>
<td>22.3</td>
<td>5.2</td>
<td><strong>60.2</strong></td>
<td>15.9</td>
<td>4.4</td>
<td><strong>54.5</strong></td>
</tr>
<tr>
<td>FIGURE</td>
<td>47.8</td>
<td>35.3</td>
<td><strong>63.5</strong></td>
<td>44.0</td>
<td>33.9</td>
<td><strong>59.4</strong></td>
</tr>
<tr>
<td>FOOTER</td>
<td>55.7</td>
<td>0.0</td>
<td><strong>69.3</strong></td>
<td>48.9</td>
<td>0.0</td>
<td><strong>59.7</strong></td>
</tr>
<tr>
<td>HEADER</td>
<td>79.7</td>
<td>0.0</td>
<td><strong>88.3</strong></td>
<td>64.8</td>
<td>0.0</td>
<td>56.6</td>
</tr>
<tr>
<td>HEADING</td>
<td>53.7</td>
<td>52.1</td>
<td><strong>66.4</strong></td>
<td>33.1</td>
<td><strong>46.0</strong></td>
<td>45.4</td>
</tr>
<tr>
<td>ITEM</td>
<td>0.0</td>
<td>33.6</td>
<td><strong>50.5</strong></td>
<td>0.0</td>
<td><strong>35.3</strong></td>
<td>33.5</td>
</tr>
<tr>
<td>ITEMIZE</td>
<td>0.0</td>
<td>25.0</td>
<td><strong>58.3</strong></td>
<td>0.0</td>
<td>25.0</td>
<td><strong>50.0</strong></td>
</tr>
<tr>
<td>KEYWORDS</td>
<td>36.4</td>
<td>0.0</td>
<td><strong>59.0</strong></td>
<td>36.4</td>
<td>0.0</td>
<td><strong>43.0</strong></td>
</tr>
<tr>
<td>PAGE NR.</td>
<td>74.7</td>
<td>0.0</td>
<td><strong>77.3</strong></td>
<td>28.5</td>
<td>0.0</td>
<td><strong>42.0</strong></td>
</tr>
<tr>
<td>TAB. CAPTION</td>
<td>55.2</td>
<td>69.1</td>
<td><strong>76.6</strong></td>
<td>40.2</td>
<td>61.6</td>
<td><strong>63.4</strong></td>
</tr>
<tr>
<td>TABLE</td>
<td>84.5</td>
<td>96.3</td>
<td>94.3</td>
<td>62.7</td>
<td>87.9</td>
<td><strong>89.6</strong></td>
</tr>
<tr>
<td>TABULAR</td>
<td>78.4</td>
<td>50.8</td>
<td><strong>100.0</strong></td>
<td>68.4</td>
<td>42.4</td>
<td><strong>99.5</strong></td>
</tr>
</tbody>
</table>

Table 3.1: Average precision (AP) of entity detection.
3.5. Results

Figure 3.3: Performance of entity detection (mAP for IoU = 0.5) during fine-tuning.

DocParser WS. For DocParser Baseline, we attribute this to the limited amount of samples in arXivdocs-target for the affected categories, coupled with an inferior model initialization, compared to DocParser WS+FT.

DocParser WS+FT outperforms the DocParser Baseline system across all measured IoU thresholds by a considerable margin. 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 3.1 breaks down the performance by entity category. For DocParser WS+FT, we observe an especially good performance for detecting tabulars and figures. This is owed to the strong initialization of our system due to the high quality and large number of samples in our scalable weak supervision.

Figure 3.3 shows the fine-tuning. Only 20 fine-tuning samples are sufficient for DocParser WS+FT to surpass the baseline system DocParser (which is trained on 160 samples from the target dataset). It thus helps in reducing the labeling effort by a factor of around 8. Furthermore, we observe a steady increase in the 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.

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 baseline and WS+FT systems.
3.5.1.2 Prediction of hierarchical relations

Table 3.2 compares the classification of relations with and without post-processing. The best performance (across all $\Psi$) is achieved by DocParser WS+FT with an IoU of 0.5: it registers an F1 score of 0.615. Here, the use of weak supervision with fine-tuning yields consistent improvements. This is also due to the significant improvements of the prior entity detection for this system variant. In particular, for an IoU of 0.5, it outperforms the F1 score of the baseline system (F1 of 0.453) by 0.162. This amounts to a relative improvement of 35.8%. Evidently, a smaller IoU threshold of 0.5 is beneficial. Higher IoU thresholds reduce the overall parsing performance as structure parsing builds on the prior detection of document entities.

The performance on hierarchical relations (F1 score of 0.615) is largely explained by our choice of a strict evaluation (i.e. the complete tuple including both entities must be correct). Overall, this performance is already highly effective in recovering the overall document structure. This is later confirmed as part of a qualitative assessment.

### 3.5.2 Robustness check: table structure parsing

**Results:** Table 3.3 compares the state-of-the-art for table structure parsing with our weak supervision strategy. Altogether, our weak supervision outperforms the state-of-the-art (Schreiber et al., 2017) by a considerable margin.

**Discussion:** Our system shows significant improvement over the image-based state of the art.
### 3.6. Related work

<table>
<thead>
<tr>
<th>System</th>
<th>Schreiber et al. (2018)</th>
<th>Baseline</th>
<th>WS</th>
<th>WS+FT</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1*</td>
<td>0.9144</td>
<td>0.8443</td>
<td>0.8117</td>
<td><strong>0.9292</strong></td>
</tr>
<tr>
<td>F1</td>
<td>—</td>
<td>0.8209</td>
<td>0.8056</td>
<td><strong>0.9292</strong></td>
</tr>
</tbody>
</table>

**Table 3.3:** ICDAR 2013 result on table structure parsing.

*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., 2017) use a different, non-public 50% random subset. Furthermore, (Schreiber et al., 2017) choose 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.

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, DocParser achieves an F1 score of 0.9292, compared to 0.9221 for the PDF-based system. We refrain from directly comparing the aforementioned F1 score with that from earlier experiments as the underlying target domains differ.

### 3.6 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 to predict the bounding boxes of tables, i.e., whether a pixel refers to a table or not (e.g., Yildiz et al., 2005; Wang et al., 2004). Prior research on table detection has utilized data augmentation (Gilani et al., 2017), weak supervision (Li et al., 2020b), and transfer learning (e.g., Siddiqui et al., 2018) to address the lack of large-scale domain-specific datasets. Similar to our research, efficient learning presents an issue for table detection. However, parsing of full pages requires effective identification of a much larger number of entities of multiple categories and high variety in shape per input.

**Table structure parsing:** There are works that recognize table structures from text or other
syntactic tokens (Kieninger and Dengel, 1998; Pivk et al., 2007) 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 works by (Schreiber et al., 2017; Qasim et al., 2019) draw upon deep neural networks to identify table structures for rendered inputs. However, they aim at a different purpose: parsing table structures, but not complete document hierarchies. As such, the authors do not attempt to identify text elements, nested figures, etc.

**Weak supervision for document layout:** (Zhong et al., 2019) use weak supervision for detection of page layout entities. The WS mechanism relies on matching external XML annotations with text extractions by a heuristic-based third-party tool. In contrast, our weak supervision directly builds on the \LaTeX\ compilation and can be readily extended to any new dataset of \LaTeX\ source files. Furthermore, the dataset features only 5 coarse categories and the system does not feature a relation classification component, thus being insufficient to acquire full document structures.\(^{10}\)

**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 (Prölllochs et al., 2019); however, not for document structure parsing.

### 3.7 Discussion

**Efficiency:** Our system requires only ~340 ms/document during entity detection (averaged over our validation set of 79 documents for DocParser WS+FT) on a single Titan Xp GPU with 12 GB VRAM and a batch size of 1. The relation detection in stage 2 only adds a minimal overhead of an average of 5.67 ms/document (10.81 ms/document with refinement) on a single CPU @ 2.1 GHz.

\(^{10}\)Additional comparison is included in the Appendix.
3.8. Summary

**Qualitative assessment:** We performed a qualitative analysis on a subset of documents. We observe that, even for F1 scores below 0.5, the final document structure is often still very accurate. In fact, state-of-the-art OCR systems as natural baselines are outperformed significantly. This can be explained by our experiment design: we used very strict evaluation metrics. Hence, even small mismatches or ambiguities between the ground truth and predicted entities result in fairly large F1 penalties, despite high overall similarity. Details are in the Appendix (including qualitative examples).

**Detection model choice:** Deep CNN models, including recent work (Tan and Le, 2019; Duan et al., 2019), are heavily reliant on large training datasets. As such, we expect the impact of our technical contribution, as shown in our comparison of baseline and WS+FT models, to be the same across different modern CNN backbones. Our choice of Mask R-CNN as a tool for instance segmentation was also done in consideration of possible future extensions of DocParser to non-rectified documents. Here, the additional instance masks could guide the OCR or rectification process.

**Future work:** In future work, we plan to explore approaches that can jointly learn entity and relation detection. Furthermore, systems could be further improved by enriching 2D inputs with textual features, e.g. high-dimensional word embeddings. The robustness of WS pretraining w.r.t. smaller subsets of arXivdocs-weak is another area of future investigation.

**3.8 Summary**

This chapter introduced DocParser, a system for full hierarchical document parsing, as well as two task-specific datasets for weak supervision, fine-tuning and evaluation. Despite the extensive interest of the NLP community in leveraging document structures e.g. (Apostolova and Tomuro, 2014; Schäfer et al., 2011; Schäfer and Weitz, 2012; Schreiber et al., 2017; Katti et al., 2018), the task of parsing complete document structures from renderings has been overlooked. To the best of our knowledge, we present the first system for this task. In particular, the combination of DocParser with text recognition systems provides an effective alternative to state-of-the-art OCR which is still widespread in practice. In addition, DocParser allows providing additional semantic input to downstream NLP tasks (e.g. information extraction).
Chapter 4

DSG: An End-to-End Document Structure Generator

This chapter is based on the manuscript (Rausch et al., 2023a), which is a joint work with Gentiana Rashiti, Maxim Gusev, Ce Zhang and Stefan Feuerriegel. Gentiana Rashiti and Maxim Gusev contributed to dataset, postprocessing, and hOCR conversion as part of their theses/projects under my supervision.

4.1 Introduction

Chapters 1 and 3 discussed the great demand for systems that can parse hierarchical document structures, but the limited systems for solving this task.

To address this, earlier research focused on custom systems for parsing specific entities in documents such as table structures (Li et al., 2020b; Smock et al., 2022) but without parsing the complete hierarchical structure in documents. Even other research aimed at identifying document entities (Antonacopoulos et al., 2009b; Zhong et al., 2019; Appalaraju et al., 2021), but without actually generating hierarchical document structures.

The system presented in Chapter 3 is tailored to generate hierarchical document structures from document renderings (Rausch et al., 2021). Yet, this work is based on heuristics and is thus not end-to-end trainable, which is why its flexibility is limited. To the best of our knowledge, there is no system for parsing hierarchical document structures that is end-to-end trainable.
Chapter 4. **DSG: An End-to-End Document Structure Generator**

### 4.1 Cross-Modal Scene Networks

We extend single-modality classification networks [24] in order to handle multiple modalities. The main modifications we introduce are that we a) have one network for each modality and b) enforce higher-level layers to be shared across all modalities. The motivation is to let early layers specialize to modality specific features (such as edges in natural images, shapes in line drawings, or phrases in text), while higher layers are meant to capture higher-level concepts (such as objects) in a representation that is independent of the modality.

#### Fig. 3: Scene Networks

We use two types of networks. a) For pixel based modalities, we use a CNN based off [46] to produce pool5. b) When the input is a description, we use an MLP on skip-thought vectors [23] to produce pool5 (as text cannot be easily fed into the same CNN).

**Figure 4.1:** The task of generating document structures by identifying (i) entities within documents and (ii) relations describing the hierarchical structure.

**Objective:** Our aim is to build an **end-to-end trainable** system that can robustly generate hierarchical document structures from document renderings.

**Our DSG system:** We develop Document Structure Generator (DSG), a system for generating hierarchical document structures from document renderings where the system is fully **end-to-end trainable**. Our DSG builds upon a deep neural network for parsing (i) entities in documents (e.g., figures, text blocks, headers, etc.) and (ii) relations that capture the sequence and nested structure between entities. In contrast to existing systems for generating document structures, our DSG uses a trainable component for classifying relations and thereby circumvents the use of heuristics. As a result, our DSG predicts entire document structures and is thus fully end-to-end trainable. This makes our system highly flexible for handling a variety of documents that can arise in practice. Finally, our DSG has a custom conversion engine to generate structured document output files in hOCR markup language, which allows for seamless integration into existing document storage and processing workflows.

We further contribute a novel, large-scale dataset for generating hierarchical document structures called E-Periodica. E-Periodica is based on real-world magazines from different source language (e.g., English, German, French, Italian). We manually annotated the hierarchical doc-
4.1. Introduction

...ument structure for several hundred magazine pages. Overall, E-Periodica contains 542 documents with more than 11,000 annotated entities. Thereby, we extend over previous datasets that have been primarily limited to scientific articles (Rausch et al., 2021). However, a limitation of scientific articles is that they follow a fairly similar structure, while magazines are characterized by large heterogeneity in their presentation and thus complex document structures. Hence, E-Periodica provides a novel and challenging, real-world setting for evaluation.

4.1.1 Contributions

Our main contributions are as follows:

1. We develop a novel system for generating hierarchical document structures from document renderings called DSG. To the best of our knowledge, our DSG is the first system for this task that is end-to-end trainable.

2. We contribute a novel, large-scale dataset called E-Periodica with manual annotations for evaluation.

3. We show that our DSG system achieves state-of-the-art performance. We further demonstrate the effectiveness of end-to-end training.

4.1.2 Overview

Section 4.2 describes the background and related work of this chapter. In Section 4.3 we specify the problem description to the tasks tackled by the system presented in this chapter. In Section 4.4 we introduce the DSG system and provide details about the individual components of the system. In Section 4.5, we describe the datasets that are used in this chapter, including modifications and annotation process. We provide details on training procedure and evaluation in Section 4.6. Section 4.7 features a qualitative and quantitative evaluation of DSG and comparison with SOTA systems. We demonstrate a set of query functionality that is enabled by the outputs of our system in Section 4.8. Finally, we discuss the findings of this chapter in Section 4.9 and provide a summary in Section 4.10.
4.2 Related work

Document structure parsing: Table 4.1 shows an overview of key systems for document structure parsing. Importantly, existing systems have two shortcomings in that they are either (i) limited to entity recognition and thus do not generate hierarchical structures, or (i) based on heuristics and thus not end-to-end trainable. We provide a detailed overview in the following.

OCR systems: Extracting text from document images has been extensively studied as part of OCR systems (Schäfer et al., 2011; Schäfer and Weitz, 2012). As such, the works focus primarily on extracting textual content, but not the hierarchical document structure, which is the objective in our research.

Entity-specific parsers: Several systems focus on specific semantic entities, namely, table detection and table structure parsing. In table detection, the task is to predict the bounding boxes of tables within document renderings, rather than generating the actual table structures (Yildiz et al., 2005; Wang et al., 2004; Gilani et al., 2017; Li et al., 2020b; Siddiqui et al., 2018). In table structure parsing, the aim is to recognize the structure (e.g., rows, cells) in tables (Smock et al., 2022; Li et al., 2020b). Here, the input is provided either as text (Kieninger and Dengel, 1998; Pivk et al., 2007) or through document renderings (Schreiber et al., 2017). However, the works are limited to a single entity (tables). Hence, these works cannot identify other entities and thus not the full document structure. Others works (e.g., (Zhong et al., 2019; Appalaraju et al., 2021)) perform entity detection in documents by locating specific elements, but again without extracting hierarchical structures. Even others focused only on the segmentation of individual lines (Joseph and George, 2021).

Hierarchical document parsers: Closest to our work is a system called DocParser (Rausch et al., 2021), which is specifically designed to capture hierarchical document structures. DocParser consists of five components (image conversion, entity detection, relation classification, structure-based refinement, and scalable weak supervision) in order to generate both entities and relations. However, in DocParser, relations are detected based on manual heuristics and are not trainable. Hence, to the best of our knowledge, systems for hierarchical document parsing that are end-to-end trainable are lacking.

Scene graph generation: Scene graph generation is a computer vision task that combines the entity detection vision task with an additional relation classification (Chang et al., 2023). Many recent methods for scene graph generation are based on two-stage training procedures where the detection components build upon Faster R-CNN (Zellers et al., 2018; Khandelwal et al., 2023).
4.3. Problem description

<table>
<thead>
<tr>
<th>System</th>
<th>Document</th>
<th>Hierarchical</th>
<th>End-to-end</th>
</tr>
</thead>
<tbody>
<tr>
<td>PubLayNet (Zhong et al., 2019)</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>DocFormer (Appalaraju et al., 2021)</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>UniDoc (Gu et al., 2021)</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>DiT (Li et al., 2022a)</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>TableBank (Li et al., 2020b)</td>
<td>only tables</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>PubTables (Smock et al., 2022)</td>
<td>only tables</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>DocParser (Rausch et al., 2021)</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>DSG (ours)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 4.1: Overview of key systems for document structure parsing from document renderings.

2021; Tang et al., 2019). However, systems for scene graph generation are predominantly used to parse real-world images and, to the best of our knowledge, have not yet been adapted to document structure parsing. This is our contribution.

Research gap: Existing systems for generating hierarchical document structures are based on heuristics and are thus not end-to-end-trainable, which limits their flexibility. As a remedy, we develop our DSG, the first system for hierarchical document parsing that is end-to-end trainable.

4.3 Problem description

Objective: The objective of our system is to generate hierarchical document structures from document renderings (e.g., PDF files, scanned images).

Our structure parsing task is analogous to that of the DocParser system in Chapter 3. Formally, the input is given by document renderings \( D_1, \ldots, D_n \). The outputs are hierarchical structures \( H_1, \ldots, H_n \). The hierarchical structure is defined by a set of (i) entities in the documents (e.g., figures, text blocks, headers, etc.) and (ii) relations that capture the sequence and nested structure between entities. Formally, entities are given by \( E_j, j = 1, \ldots, m \), and relations by \( R_j, j = 1, \ldots, k \). We refer to Section 2.1 for a detailed description of entities and relations.

hOCR output: As additional output, the hierarchical document structures \( H_i, i = 1, \ldots, n \), should further be paired with recognized texts \( T_i, i = 1, \ldots, n \) and provided as marked-up text files \( M_i, i = 1, \ldots, n \) in hOCR format (Breuel, 2007) to facilitate downstream processing task. hOCR is a markup language for representing and storing structured documents in a unified
format. Hence, this should ensure that the output can be directly used by common tools for document processing and storage workflows that are widespread in practice.

4.4 Our DSG system

Overview: An overview of our DSG system is shown in Fig. 4.2. The objective of our DSG is to generate hierarchical document structures from document renderings in an end-to-end trainable setup. For this, our system builds upon a deep neural network that consists of fully trainable components to parse both entities and subsequently the relations that represent the hierarchical structures. Our system processes documents along five components: (C1) image preprocessing, (C2) entity detection, (C3) relation classification and entity refinement, (C4) grammar-based postprocessing, and (C5) hOCR conversion engine. The components are described in the following.

4.4.1 Image preprocessing (C1)

Our system processes all input documents as rendered images. For source formats such as PDF, we first generate images for each document page, which are then used as input to our system. Images are resized bilinearly so that their smallest side has a maximum size of \( \phi_s^{\text{max}} \). If the longest side would exceed a predefined maximum size of \( \phi_l^{\text{max}} \) after this step, the images are resized so that their longest side length is \( \phi_l^{\text{max}} \). During training, the image size can be varied for the purpose of data augmentation.\(^1\) For this, \( \phi_s^{\text{max}} \) and \( \phi_l^{\text{max}} \) are randomly chosen from a set of different sizes. The images are then normalized following the procedure in (Wu et al., 2019). Specifically, we subtract the mean channel-wise pixel values of the underlying pre-training dataset (Deng et al., 2009) from the inputs.

\(^1\)In computer vision, it is common to perform additional data augmentations through image mirroring or rotation operations, which are commonly applied to facilitate training and system performance. However, we avoid such data augmentations in our work because the hierarchical document structures are sensitive to the original document geometry (e.g. left-to-right reading orders).
4.4. Our DSG system

4.4.1 Cross-Modal Scene Networks

We extend single-modality classification networks [24] in order to handle multiple modalities. The main modifications we introduce are that we a) have one network for each modality and b) enforce higher-level layers to be shared across all modalities. The motivation is to let early layers specialize to modality-specific features (such as edges in natural images, shapes in line drawings, or phrases in text), while higher layers are meant to capture higher-level concepts (such as objects) in a representation that is independent of the modality.

![Diagram of DSG system]

**Figure 4.2:** Overview of our DSG system.

4.4.2 Entity detection (C2)

The second component builds upon a Faster R-CNN architecture (Ren et al., 2015) for entity detection. Here, visual feature maps on different scales are extracted via a convolutional neural
network (He et al., 2016; Lin et al., 2017). The visual feature maps are then passed on to another network component, called region proposal network (RPN), which generates a set of rectangular candidate entity region proposals in the image. For each of the region proposals, a category prediction network is applied to predict the semantic category $c_j$ of an entity $E_j$. If the confidence score $P'_j$ of a candidate region surpasses a predefined threshold, it is accepted as an entity (and discarded otherwise). Subsequently, an additional neural network is used to predict the size and position of the initial rectangular region $B_j$ (based on the rectangular candidate entity region proposals from the RPN). Afterward, the entities are passed on to component C3, which is responsible for the relation classification.

### 4.4.3 Relation classification and entity refinement (C3)

The relation classification in DSG builds upon the neural motifs architecture (Zellers et al., 2018). This architecture extends the entity detection architecture with two additional neural network heads. Concretely, the detected entities are passed on to neural network heads for relation classification and entity refinement. In the following, we refer to these two heads as the relation head and the refinement head, respectively.

Both the relation head and the refinement head build on bidirectional long short-term memory (LSTM) networks that take the entities from component C2 as input. Both proceed in slightly different ways. The relation head is fed with pairs of subject entity and object entity, $(E_{subj}, E_{obj})$, to classify if they form a relation triple $(E_{subj}, E_{obj}, \Psi)$ of type $\Psi \in \{parent\_of, followed\_by, null\}$. The refinement head maps the categorical labels $c_j$ and their confidence scores $P_j$ from component C2 onto refined categories $c_j$ and confidence scores $P_j$ by taking into account the contextual information of all predicted entities. Formally, both relation head and refinement head are implemented as follows:

i  **Relation head:** The relation classification returns a confidence score $P_j^\Psi$ for all considered entity pairs $(E_{subj}, E_{obj})$. If the respective confidence score exceeds a predefined threshold $\tau$, a relation $R_j$ of type $\Psi \in \{parent\_of, followed\_by, null\}$ is accepted. Here, the relation type $\Psi = \text{null}$ is used to indicate the absence of a hierarchical relation.

ii  **Refinement head:** The refinement head is fed with additional features $\rho_{vis}^{\text{ref, in}}, i \in \{\text{vis, cat, pos}\}$, as follows. First, $\rho_{vis}^{\text{ref, in}}$ refers to the visual feature map that is extracted by the underlying Faster R-CNN architecture and corresponds to the image region of the entity in the rendered document. Second, $\rho_{cat}^{\text{ref, in}}$ is a category embedding, which is based on a pre-trained
word embedding dictionary and is selected according to the predicted semantic category of the entity (Pennington et al., 2014). Third, $\rho_{pos, i}^{ref, in}$ is a positional embedding to represent the size and location of the entity bounding box. Specifically, the positional embedding incorporates the width, height, and location of the bound box $B_j$. We refer later refer to the features $\rho_i^{ref, in}, i \in \{vis, cat, pos\}$ as refinement context input features.

The refinement context input features are passed to the LSTM from the refinement head and, subsequently, a fully connected layer to produce so-called refinement context output features $\rho_{ref, out}^{ref, in}$. These features are then used to predict the refined entity categories $c_j$.

Subsequently, the relation head is fed with three relation context input features $\rho_i^{rel, in}, i \in \{vis, cat, ref\}$ as follows. First, $\rho_{vis}^{rel, in}$ is visual feature map, identical to that in $\rho_{vis}^{ref, in}$. Second, $\rho_{cat}^{rel, in}$ is the category embedding, analogous to the category embedding $\rho_{cat}^{ref, in}$ to represent the refined entity category. Third, $\rho_{ref}^{rel, in}$ are the refinement context output features, i.e., $\rho_{ref}^{rel, in} = \rho_{ref, out}^{ref, in}$.

Next, pairs of two entities are processed by the relation head. Crucially, unlike existing systems such as DocParser, this step is fully trainable. For this, the relation head forms three so-called pair-wise features $\rho_1^{pair}, \rho_2^{pair},$ and $\rho_3^{pair}$ as specified in the following. Later, the pair-wise features are used to predict confidence scores $P_j$ for all considered entity pairs ($E_{subj}, E_{obj}$). Specifically, the pair-wise features are: First, $\rho_1^{pair}$ is the visual feature map that is extracted by the Faster R-CNN architecture. It corresponds to the image region of the entity pair $B_{pair} = \text{Union}(E_{subj}, E_{obj})$. Second, $\rho_2^{pair}$ is formed for entity-entity pairs ($E_{subj}, E_{obj}$) by concatenating the respective refinement context output features ($\rho_{subj}^{rel, out}, \rho_{obj}^{rel, out}$). Third, a frequency bias term $\rho_3^{pair}$ is calculated from the refined categories $c$ of the pair-wise considered entities. The frequency bias term is based on the empirical distribution over relations ($E_{subj}, E_{obj}, \Psi$) in the training set. The frequency bias term thus reflects that, for certain pairings of entity categories ($c_{subj}, c_{obj}$), relation types $\Psi \in \{parent_of, followed_by, null\}$ are more or less likely. Finally, the pair-wise features $\rho_1^{pair}, \rho_2^{pair},$ and $\rho_3^{pair}$ are then combined to a pair-wise output feature $P_{out}^{pair}$ that used to predict the $P_j$ for all entity pairs.

As a result, the entire component for relation classification is end-to-end trainable. This is crucial difference of our DSG over existing systems.
4.4.4 Grammar-based postprocessing (C4)

This component of our system converts hierarchical document structures \( H_i, i = 1, \ldots, n \), consisting of the predicted entities \( E \) and relations \( R \), into a postprocessed document structure \( H' \). Here, the aim is to ensure a valid, tree-structured format that can later be used to generate different output formats such as hOCR (Breuel, 2007). For this, we ensure that all entities form a tree structure w.r.t. their hierarchical relations of type \( \Psi = \text{parent.of} \) and are connected to a root entity with \( c = \text{doc.root} \). We note that our postprocessing does not make any assumptions about the geometric overlap or the document layout. To this end, it is purely based on the document grammar and the predicted confidence scores \( P^p \). In particular, we apply our grammar-based postprocessing in sequential steps to address root entities \( (g_{r}) \), illegal entities \( (g_{ilg}) \), and missing relations \( (g_{mis}) \) as follows:

- **Root entities** \( (g_{r}) \): We append additional entities to build a basic skeleton for the document files. Specifically, we add root entities \text{doc.root}, \text{article} and \text{meta}. To enable full end-to-end training, we allow the prediction of these entities in the training process.

- **Illegal relations** \( (g_{ilg}) \): During training and inference of the relation classification, no restrictions are made on the possible combinations of entity pairs. This choice is made to allow for flexibility during end-to-end training (e.g., in the event that entities are not correctly predicted by the detection component of DSG). However, such flexibility can result in relations that violate our document grammar and where thus conflicts must be resolved. For example, we remove potential cycles so that the hierarchical document structure forms a tree structure. Details are in the Appendix.

- **Missing relations** \( (g_{mis}) \): Relations are added in order to ensure a valid tree structure so that each entity has a valid relation of the type \( \Psi = \text{parent.of} \). Specifically, every entity must have exactly one parent, except for the entity with \( c = \text{doc.root} \), which has none. If an entity does not have a parent, we add a corresponding relation \((E_{\text{subj}}, E_{\text{obj}}, \Psi)\) with \( E_{\text{obj}} = E \) and \( \Psi = \text{parent.of} \). Details are in the Appendix.

4.4.5 hOCR conversion engine (C5)

In DSG, the final component is an hOCR conversion engine. It takes a postprocessed document structure \( H' \) as input and then converts it into a hOCR file that is compatible with common open-
source tools for document processing workflows. Below, we extend the common hOCR format (Breuel, 2007) to additionally accommodate hierarchical structures.

The hOCR format (Breuel, 2007) encodes information using extensible markup language (XML) and is built upon hypertext markup language (HTML). To ensure compatibility with standard hOCR tools while still accommodating the hierarchical structure from DSG, we add a new DSG-specific XML node `<div>` into the standard hOCR XML nodes. Our new node does not have a hOCR-specific class attribute. As a consequence, third-party tools are still able to process our output as valid hOCR, since only hOCR-specific class attributes are considered by default. In plain words, the hierarchy and sequential ordering are preserved, while additional information from our DSG, such as the extended set of semantic categories, is ignored.

We use specific hOCR elements to convert the postprocessed hierarchical structure $H'$ into an hOCR file. We use the following mapping to create the hOCR XML nodes from DSG entities. Formally, we specify a mapping $\omega : c \rightarrow \eta$ of the DSG entity categories $c$ to the hOCR element classes, denoted by $\eta$. We provide a list of $(c, \eta)$ tuples in Table 4.2. The hOCR format does not have elements that match the semantic categories of meta and article in DSG. In order to deal with both semantic categories, we keep the DSG-specific XML nodes to preserve the underlying structural information.

<table>
<thead>
<tr>
<th>DSG semantic cat.</th>
<th>hOCR class</th>
<th>DSG semantic cat</th>
<th>hOCR class</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOC. ROOT</td>
<td>ocr_page</td>
<td>ITEM</td>
<td>ocr_carea</td>
</tr>
<tr>
<td>META</td>
<td>None</td>
<td>ITEMIZE</td>
<td>ocr_float</td>
</tr>
<tr>
<td>AUTHOR</td>
<td>ocr_author</td>
<td>ORDERED GROUP</td>
<td>ocr_carea</td>
</tr>
<tr>
<td>BACKGROUND FIG.</td>
<td>ocr_float</td>
<td>PAGE NR.</td>
<td>ocr_pageno</td>
</tr>
<tr>
<td>TEXT BLOCK</td>
<td>ocrx:block</td>
<td>TABLE</td>
<td>ocr_table</td>
</tr>
<tr>
<td>FIGURE</td>
<td>ocr_float</td>
<td>TABULAR</td>
<td>ocr_table</td>
</tr>
<tr>
<td>FIGURE GRAPHIC</td>
<td>ocr_photo</td>
<td>TABLE OF CONTENTS</td>
<td>ocr_table</td>
</tr>
<tr>
<td>FIGURE CAPTION</td>
<td>ocr_caption</td>
<td>UNORDERED GROUP</td>
<td>ocr_float</td>
</tr>
<tr>
<td>FOOTER</td>
<td>ocr_footer</td>
<td>ARTICLE</td>
<td>None</td>
</tr>
<tr>
<td>FOOTNOTE</td>
<td>ocr_footer</td>
<td>COLUMN</td>
<td>ocr_carea</td>
</tr>
<tr>
<td>HEADER</td>
<td>ocr_header</td>
<td>ROW</td>
<td>ocr_carea</td>
</tr>
<tr>
<td>HEADING</td>
<td>ocr_header</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.2: Mapping between semantic categories in DSG and hOCR classes.

The conversion process consists of three steps to convert the postprocessed hierarchical structures $H'$ into hOCR files:
1. **Initialization** ($s_1$): We initialize the hOCR file with the hierarchical structure $H'_{\text{parent}}$ that only considers relations of type $\Psi = \text{parent}$. For this, hOCR and additional DSG XML nodes are initialized according to $H'_{\text{parent}}$ and the category mapping $\omega$.

2. **Order of children** ($s_1$): We ensure a correct ordering of the children. Formally, we ensure that, if any $\Psi = \text{followed by}$ relation exists between two entities in $H'$, the corresponding XML nodes follow that order.

3. **OCR enrichment** ($s_3$): We additionally enrich the hOCR files with the textual contents $T$ of the documents. First, we ensure that words are only appended to leaf node entities $E_{\text{leaf}}$ in the document structure. Second, words are assigned to the entity $E_{\text{leaf}}$ with the highest intersection-over-union score (see Sec. 4.6.1 between entity bounding box $B_{\text{leaf}}$ and word bounding box $B_{\text{word}}$).

A key feature of the hOCR format is the ability to perform structure-based XPath queries on document files (Breuel, 2007). To facilitate such applications in practice, we extend the XPath queries to account for the hierarchical structures in our hOCR files generated by DSG. As a result, we allow that hOCR files can be searched for specific DSG entities and relations using XPath queries. We provide XPath queries for three different types of queries:

1. **Node name search** ($q_1$): This is a simple search by node name that returns all XML nodes that match the desired node name.

2. **Absolute path search** ($q_2$): We offer the ability to navigate documents using absolute paths. An absolute path starts with a / symbol and then describes the path to the desired node starting from the root node of the XML file. Here, the different names of the nodes along this path are concatenated using / symbols.

3. **Relative path search** ($q_3$): Relative paths can be used to retrieve a desired node. For this, two wildcard elements are used to match any potential node or attribute. The starting symbol // indicates a relative path search. The relative path search then returns any node that has a path that could potentially match the query by using two wildcard elements * to match any node or path and @ to match any node attribute.

### 4.4.6 Implementation details

Our DSG system is based on the neural motifs architecture (Zellers et al., 2018) using the implementations from (Wu et al., 2019; Khandelwal et al., 2021). However, we make non-trivial
adaptations to accommodate our task of generating hierarchical document structures, as detailed in the following.

**Image preprocessing (component C1):** Images are bilinearly resized so that their smallest side has at most size $\phi_s^{\text{max}}$. If the longest side would exceed a predefined maximum size of $\phi_l^{\text{max}}$ after this step, resizing is instead done so that the longest side length is $\phi_l^{\text{max}}$. During training, the image size is varied for augmentation purposes. For this, $\phi_s^{\text{max}}$ is chosen randomly from a set of different sizes. We set $\phi_l^{\text{max}}$ at 600, while $\phi_s^{\text{max}}$ is randomly chosen from the range [250, ..., 550] using increments of 50. During training, images are randomly resized by applying the aforementioned resizing scheme. For testing, we set $\phi_s^{\text{max}} = 400$ and $\phi_l^{\text{max}} = 600$.

**Entity detection (component C2):** The entity detection component is based on the Faster R-CNN architecture (Ren et al., 2015). We refer the reader to the general architecture overview in Figure 2.1. We use a ResNet (He et al., 2016) backbone of depth 50 in our system. Training of component C2 uses the loss term $L_{C2} = L_{\text{cls}}^{\text{RPN}} + L_{\text{loc}}^{\text{RPN}} + L_{\text{cls}}^{\text{E}} + L_{\text{loc}}^{\text{E}}$. The losses $L_{\text{loc}}^{\text{RPN}}$ and $L_{\text{cls}}^{\text{RPN}}$ penalize localization and classification errors for the candidate regions generated by the region proposal network (RPN). Furthermore, the objective of correct localization and classification of predicted entities is formulated via the losses $L_{\text{loc}}^{\text{E}}$ and $L_{\text{cls}}^{\text{E}}$, respectively. During training, up to 50 entities are passed from the entity detection component (C2) to the component responsible for relation classification and entity refinement (C3).

**Relation classification (component C3(i)):** The relation head uses a bidirectional LSTM with one recurrent layer, a hidden layer size of 512, and a dropout of 0.2. The relation context input features $\rho_{1}^{\text{rel}}$ are extracted via the underlying Faster R-CNN architecture for the bounding box $B_j$ of each entity. Specifically, the convolutional neural network (He et al., 2016) of our Faster R-CNN architecture processes the input images in multiple sequential steps with decreasing spatial resolution. The output features of the Faster R-CNN are then fed into a feature pyramid network (FPN) (Lin et al., 2017), which produces multi-scale visual feature maps. Four multi-scale visual feature maps that correspond to the image region of $B_j$ are filtered by an alignment layer, concatenated, and passed through two fully-connected layers with ReLu activations (Nair and Hinton, 2010) to produce feature vectors of dimension 1024. These feature vectors are then used as the relation context input feature $\rho_{\text{vis}}^{\text{rel}}$.

Following (Zellers et al., 2018), we extract visual feature maps $\rho_{1}^{\text{pair}}$ for the union bounding box $B_{\text{pair}}$ of subject-object pairs ($E_{\text{subj}}, E_{\text{obj}}$). The extraction proceeds analogously to $\rho_{\text{vis}}^{\text{rel}}$ but uses the region $B_{\text{pair}}$ to filter the multi-scale visual feature maps, resulting in pair-wise features $\rho_{1}^{\text{pair}}, \rho_{1}^{\text{pair}}$ and $\rho_{2}^{\text{pair}}$ are fed into fully-connected layers $W_{\text{pair}}$ with output dimensions.
of 4096 and then combined using element-wise multiplication. The resulting feature vector is
finally fed into a fully-connected layer $W_{\text{pair}}$ with an output dimension equal to the number of
relation types and added to the frequency bias term $\rho_{3}^{\text{pair}}$, resulting in the pair-wise output feature
$\rho_{\text{out}} = W_{\text{rel}}( (W_{1}^{\text{pair}} \rho_{1}^{\text{pair}}) \circ (W_{2}^{\text{pair}} \rho_{2}^{\text{pair}})) + \rho_{3}^{\text{pair}} \cdot \rho_{\text{out}}$ is then used to predict the class probabilities
for the relations. For training and evaluation, the ground-truth relation triples are matched to the
candidate triples by calculating the IoU (see Sec. 4.6.1) scores $S_{\text{subj}} = \text{IoU}(E_{\text{subj}}, E_{\text{GT}}^{\text{subj}})$ and $S_{\text{obj}} = \text{IoU}(E_{\text{obj}}, E_{\text{GT}}^{\text{obj}})$ of the subject and object entities in the relation, respectively. In accordance
with our objective of uniquely matching all ground-truth relations, we allow only one candidate
relation to be considered per ground-truth relation.

Entity refinement (component C3(ii)): The refinement head is based on a bidirectional LSTM
with one recurrent layer, a hidden layer size of 512, and a dropout of 0.2. The inputs to the LSTM
are ordered according to their $x$-coordinates (center point) from left to right. The additional
refinement context input features are computed as follows. The visual feature map $\rho_{\text{vis}}^{\text{ref}}$ is
computed analogously to $\rho_{\text{vis}}^{\text{rel}}$. The category embedding $\rho_{\text{cat}}^{\text{ref}}$ is computed by mapping the
name of the semantic category directly onto the GloVe word embedding with an identical name
(Pennington et al., 2014). For some categories that are encountered in the our datasets, there
is no direct match. For these categories, we use the following mapping: (BIBLIOGRAPHY BLOCK
$\leftrightarrow$ bibliography), (TEXT BLOCK $\leftrightarrow$ paragraph), (FIGURE CAPTION $\leftrightarrow$ caption), (FIGURE GRAPHIC $\leftrightarrow$
graphic), (PAGE NR. $\leftrightarrow$ numbering), (TABLE CAPTION $\leftrightarrow$ caption). The word embedding dimension
is set to 200 in our experiments. For the positional embedding $\rho_{\text{pos}}^{\text{ref}}$, the bounding box width
and $x$-coordinates are normalized with respect to the width of the full-sized image. Analogously,
we normalize the box height and the $y$-coordinates with respect to the height of the full-sized
image.

Training of component C3 uses the loss term $L_{C3} = L_{\text{ref}} + L_{\text{rel}}$, consisting of losses of relation
classification, $L_{\text{rel}}$ and class refinement, $L_{\text{ref}}$.

4.5 Datasets

We compare our system using two different datasets; see Table 4.3. Datasets that are suitable
for evaluation must contain annotations for the full hierarchical document structure, and, hence,
existing datasets are so far limited to scientific articles (Rausch et al., 2021). However, scientific
articles follow a fairly similar structure. To this end, we also introduce a new dataset called
E-Periodica containing real-world, offline magazines. This is beneficial for our evaluation as
it provides a dataset with large heterogeneity in the presentation and thus complex document structures. The datasets are described in the following.

### 4.5.1 arXivdocs-target

The existing dataset for training and evaluation, arXivdocs-target, contains 362 hand-annotated documents (Rausch et al., 2021). Previously, the dataset has only been used for training entity detection and not relation classification. To this end, we perform the following processing steps to allow end-to-end training. We first split any multi-page documents into separate single-page images. We then convert the dataset to a standardized format (Krishna et al., 2017) to allow processing with standardized benchmark libraries and document parsing codebases.

### 4.5.2 E-Periodica

E-Periodica is a project that aims to digitize a wide range of historical and contemporary magazines for future generations (Wanger, 2018). It comprises magazines from different source languages such as English, German, French, and Italian. Magazine pages often have a complex structure and follow little consistency in the formatting rules as compared to scientific literature. Examples are shown in Figures 4.3 and 4.4. The entire E-Periodica contains over 8 million pages from over 400 journals. We manually annotated 542 documents comprising 11,446 annotated entities. Specifically, we sampled a subset of document pages from journal issues of the past six decades. Moreover, the distribution of pages per journal is highly irregular, and, for this reason, we only consider five pages per issue of a given magazine in a given year. We then annotated entities and the relations between all entities to form hierarchical document structures. Details on the annotation procedure are provided below. We split the dataset into training, validation, and test sets of 270, 135, and 137 samples, respectively.

**Annotation procedure:** Our manual annotation followed a two-step process: (1) entity annotation and (2) relation annotation. In the entity annotation step, a bounding box is drawn around

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Document type</th>
<th>#Docs</th>
<th>#Semantic cat.</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>arXivdocs-target</td>
<td>Scientific articles</td>
<td>362</td>
<td>21</td>
<td>(Rausch et al., 2021)</td>
</tr>
<tr>
<td>E-Periodica</td>
<td>Magazines</td>
<td>542</td>
<td>22</td>
<td><em>Ours</em></td>
</tr>
</tbody>
</table>

*Table 4.3:* Overview of existing datasets for our task.
the entities on a page, and a semantic category is assigned to each entity. For the relation annotation step, we first annotate relations to define the reading order of a page by focusing on \( \Psi = \text{followed}_by \). If entities are nested, we additionally annotate relations that characterize nested structures given by \( \Psi = \text{parent}_of \) (e.g., (figure, figure caption, parent_of)).

Specific considerations are made for the annotation of hierarchical structures in E-Periodica. Unlike scientific documents, many magazine pages lack a standardized reading order (e.g., separate articles within a magazine can be read in arbitrary order). To model this, we designate two semantic categories for this purpose: unordered group and ordered group. An unordered group refers to parts that do not belong to the general document-level reading order (e.g., advertisements). An ordered group refers to parts that belong to the regular reading order (e.g., single column with a separate article). Further details are in the Appendix.
Summary statistics: Table 4.4 lists the semantic categories and their corresponding frequency in the E-Periodica dataset. As can be seen from the table, the semantic categories are highly diverse, especially compared to scientific articles (Rausch et al., 2021), implying that E-Periodica is a challenging dataset for benchmarking. The distribution of leaf nodes per document in Fig. 4.5 further emphasizes the complexity of our dataset due to the deeply nested structure.
### Table 4.4: Distribution of semantic categories in E-Periodica

<table>
<thead>
<tr>
<th>Category</th>
<th>Frequency</th>
<th>%</th>
<th>Avg. depth</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARTICLE</td>
<td>651</td>
<td>5.69</td>
<td>2.00</td>
</tr>
<tr>
<td>AUTHOR</td>
<td>226</td>
<td>1.97</td>
<td>3.96</td>
</tr>
<tr>
<td>BACKGROUND FIG.</td>
<td>46</td>
<td>0.40</td>
<td>5.00</td>
</tr>
<tr>
<td>COLUMN</td>
<td>228</td>
<td>1.99</td>
<td>5.98</td>
</tr>
<tr>
<td>TEXT BLOCK</td>
<td>1469</td>
<td>12.83</td>
<td>4.01</td>
</tr>
<tr>
<td>DOCUMENT ROOT</td>
<td>542</td>
<td>4.74</td>
<td>1.00</td>
</tr>
<tr>
<td>FIGURE</td>
<td>550</td>
<td>4.81</td>
<td>4.07</td>
</tr>
<tr>
<td>FIGURE CAPTION</td>
<td>196</td>
<td>1.71</td>
<td>5.06</td>
</tr>
<tr>
<td>FIGURE GRAPHIC</td>
<td>516</td>
<td>4.51</td>
<td>5.08</td>
</tr>
<tr>
<td>FOOTER</td>
<td>158</td>
<td>1.38</td>
<td>3.00</td>
</tr>
<tr>
<td>FOOTNOTE</td>
<td>59</td>
<td>0.52</td>
<td>4.00</td>
</tr>
<tr>
<td>HEADER</td>
<td>158</td>
<td>1.38</td>
<td>3.00</td>
</tr>
<tr>
<td>HEADING</td>
<td>1275</td>
<td>11.14</td>
<td>4.01</td>
</tr>
<tr>
<td>ITEM</td>
<td>1052</td>
<td>9.19</td>
<td>5.04</td>
</tr>
<tr>
<td>ITEMIZE</td>
<td>144</td>
<td>1.26</td>
<td>4.04</td>
</tr>
<tr>
<td>META</td>
<td>416</td>
<td>3.63</td>
<td>2.00</td>
</tr>
<tr>
<td>ORDERED GROUP</td>
<td>1132</td>
<td>9.89</td>
<td>3.00</td>
</tr>
<tr>
<td>PAGE NR.</td>
<td>368</td>
<td>3.22</td>
<td>3.02</td>
</tr>
<tr>
<td>ROW</td>
<td>1460</td>
<td>12.76</td>
<td>6.00</td>
</tr>
<tr>
<td>TABLE</td>
<td>124</td>
<td>1.08</td>
<td>4.00</td>
</tr>
<tr>
<td>TABLE OF CONTENT</td>
<td>89</td>
<td>0.78</td>
<td>2.00</td>
</tr>
<tr>
<td>TABULAR</td>
<td>124</td>
<td>1.08</td>
<td>4.99</td>
</tr>
<tr>
<td>UNORDERED GROUP</td>
<td>463</td>
<td>4.05</td>
<td>3.00</td>
</tr>
</tbody>
</table>
Figure 4.5: Leaf nodes in dataset.
4.6 Experimental Setup

4.6.1 Performance metrics

We separately evaluate the performance of our system for (i) entity detection and (ii) structure generation in which the hierarchical relations are considered. To this end, we adapt the benchmarking in related tasks from scene graph generation but adapt for our purpose of parsing document structures.

**Entity detection:** We follow common practice (Krishna et al., 2017; Lin et al., 2014) in benchmarking entity detection. Specifically, we determine how many entities are correctly predicted out of all ground-truth entities. We compare the predicted entities \( E_j = (c_j, B_j, P_j) \) with the ground-truth entities, consisting of the true category \( \hat{c}_j \) and the true bounding box \( \hat{B}_j \). Here, we measure the overlap between bounding boxes of the same category (Everingham et al., 2009). Specifically, we calculate the so-called intersection-over-union (IoU) via Equation 2.1. Predicted entities are considered a true positive if their IoU is higher than a pre-defined threshold. If more than one entity exceeds that threshold for the same ground-truth entity, the entity with the highest IoU is considered as a true positive. Any unmatched predicted entities and ground-truth entities are considered false positives and false negatives, respectively. We compare IoU thresholds of 0.5 and 0.75. All computations of the IoU are based on the API in (Lin et al., 2014).

We further calculate the average precision (AP) per semantic category \( c_k \in C \). The overall performance across all categories is given by the mean average precision (mAP) with (0: worst, 100: best).

4.6.2 Training and hyperparameter tuning

**Initialization:** We initialize our systems with two pre-training steps. First, the systems are initialized with weights of a Faster R-CNN architecture trained on the COCO dataset (Lin et al., 2014) with copy-paste augmentation (Ghiasi et al., 2021). Second, since the COCO dataset does not contain documents, we then proceed to pre-train all systems on arXivdocs-weak (Rausch et al., 2021). Like arXivdocs-target, this dataset has been generated for scientific articles, but it was only annotated with a weak supervision mechanism. This allows for better preparation of the document parsing tasks. The resulting system weights are then used as a starting point for our experiments.
4.6. Experimental Setup

Training: We first sample 128 from all possible entity-entity pairs per training iteration to serve as input to the relation classification of the relation head. This reduces the computational complexity of the training and allows us to sample a more balanced set of positive and negative sample, since the majority of entity-entity pairs correspond to relations of type $\Psi = \text{null}$. Unlike common training procedures from scene graph generation, we do not apply geometric constraints on candidate pairs $(E_{\text{subj}}, E_{\text{obj}})$. Concretely, this means that entity-entity pairs with no geometric overlap are considered for relation prediction. This is especially important for relations of type $\Psi = \text{followed\_by}$, where the bounding boxes $(B_{\text{subj}}$ and $B_{\text{obj}}$) of $(E_{\text{subj}}$ and $E_{\text{obj}})$ do not intersect. In order to allow for training of the refinement and relation head throughout the whole training procedure, we append any missing ground-truth entities to the set of entities that are passed to the refinement head. This is to avoid cases where no entities or only erroneous entities are detected and where thus no positive learning samples can be provided. This happens, for example, at the beginning of the training procedure.

We train our DSG system end-to-end via a joint objective consisting of (i) the entity detection component based on the Faster R-CNN component and (ii) the relation classification and entity refinement component. The losses $L_{C2}$ and $L_{C3}$ for both components are combined to train our DSG system. Our system is trained for up to 200,000 iterations with a batch size of 4 and a learning rate of 0.001. We apply early stopping based on the performance for entity detection on the validation set.

Computational performance: We measure the computational performance of our system on a machine with a single NVIDIA Titan Xp GPU with a memory size of 12GB. Here, the average time to process a sample, as measured on the validation set with a batch size of 1, is $\sim 0.1616$ seconds. Integrating our grammar-based postprocessing comes with only a small overhead and results in an average joint processing time of $\sim 0.1776$ seconds for the same computational setup.

4.6.3 Baselines

We benchmark our proposed DSG against state-of-the-art systems for document structure parsing.

- **DocParser** (Rausch et al., 2021): We reimplement DocParser (Rausch et al., 2021), which is a state-of-the-art system for hierarchical structure parsing. To the best of our knowledge, this is the only suitable baseline that can parse entire document structures (see Sec. 4.2).
To ensure comparability between DocParser and our system, we use the Faster R-CNN architecture with a ResNet (He et al., 2016) backbone of depth 50 for entity detection. Of note, DocParser uses heuristics for the relation classification and is thus not end-to-end trainable. Therefore, we evaluate DocParser using the original heuristics in (Rausch et al., 2021) for relation classification. We extend the heuristics to accommodate the additional root entity article (i.e., map it onto the entity category document).

We further compare different variants of DSG that act as ablation studies to demonstrate the importance of end-to-end training. Thereby, we can evaluate the importance of end-to-end vs. 2-stage training.

- DSG 2-stage (w/ C2 frozen): In the first stage, component C2 of DSG is trained exclusively with respect to the correct prediction of entities. In the second stage, component C3 is added to DSG training. However, the weights of component C2 are frozen during the second stage. Hence, the loss to update C3 is only based on the predictions of the relation head and refinement head.

- DSG 2-stage (w/ C2 unfrozen): In the first stage, component C2 of DSG is trained. Here, a loss is used which only learns against the correct prediction of entities. In the second stage, we allow parameter weight updates to both C2 and C3 of the system. Here, a loss is used which only learns against the predictions of the relation head and refinement head of component C3, but not against the prediction of entities by component C2.

- DSG end-to-end (w/o postprocessing): This is our DSG system from above that is trained in an end-to-end fashion but without the postprocessing from component C4.

- DSG end-to-end (w/ postprocessing): This is our DSG system from above that is trained in an end-to-end fashion.

Training procedure for baselines: We use identical hyperparameter settings for training the entity detection component that are used for the baseline systems DocParser, DSG 2-stage, and DSG end-to-end. All systems are trained for up to 100,000 iterations with a learning rate of 0.01 and a batch size of 8. This training only includes the prediction of document entities through component C2 and uses the loss term $L_{C2}$. For DocParser, we apply early stopping based on the mAP score for an IoU threshold of 0.5 on the validation set.

After training the component C2 for entity detection, we continue with the second training stage of DSG 2-stage (w/ C2 frozen) and DSG 2-stage (w/ C2 unfrozen), where the systems are
4.6. Experimental Setup

trained with a joint objective for relation classification and entity refinement using the loss term \( L_{C3} \). We perform the second stage with a learning rate of 0.001 and a batch size of 8 for up to 100,000 iterations. Unlike DSG, the systems in the ablation study do not use the additional loss \( L_{C2} \) for entity detection via component C2 in this stage. We apply early stopping based on the mAP score for an IoU threshold of 0.5 on the validation set in the second stage.

**Structure generation:** To evaluate how well the hierarchical structure is generated, we perform an evaluation of triplets for the relations. Specifically, we measure exact matches of the predicted relations \((E_{\text{subj}}, E_{\text{obj}}, \Psi)\) against the ground-truth observations and report the corresponding F1 score. The F1 score is the harmonic average of precision and recall for predicting these triples, i.e., \( F_1 = 2 \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \) with (0: worst, 1: best). For this, we first determine all matches of bounding boxes of entities. Subsequently, a relation is considered a match if the predicted relation type and both bounding boxes match with a ground-truth triple.

Our above performance measures are fairly strict when compared with evaluations in conventional scene graph generation (e.g., (Zellers et al., 2018; Khandelwal et al., 2021)). Recall that we consider a match if and only if the relation type and both bounding boxes match with a single ground-truth triplet. In contrast, conventional scene graph generation uses a relaxed definition where a predicted relation triple \((E^p_1, E^p_2, \phi^p)\) is considered a match with a ground-truth triple \((E^g_1, E^g_2, \phi^g)\) if \( \phi^p = \phi^g \) and if an IoU overlap is found between the ground-truth entities and both predicted entities. However, this definition allows for that more than one predicted entity could be matched with a ground-truth entity during evaluation of relation prediction. In our evaluation, we apply a more strict performance that considers at most one unique match with ground-truth entity, following the same procedure as during entity detection evaluation.

Our choice of performance metrics is important to effectively differentiate between closely nested entities. This is relevant, for example, to distinguish between a figure that is wrapped around a subfigure. Using simple IoU matching, a predicted entity could be matched with either of the two ground-truth candidate entities (i.e., the figure and the subfigure). However, to reconstruct the hierarchical document structure, it is crucial to correctly determine the exact hierarchical relations among entities to arrive at a unique and valid tree structure.
4.7 Results

4.7.1 Numerical Experiments

The objective of our experiments is to confirm the effectiveness of our DSG in generating the hierarchical document structures. Hence, we proceed two-fold: (1) We first measure the performance in entity detection, and (2) we measure the performance correctly generating hierarchical structures.

Entity detection: We report the performance for entity detection in Table 4.5 (arXivdocs-target) and Table 4.6 (E-Periodica). Here, we report the results for our DSG (last rows). We further state the results for DocParser (Rausch et al., 2021), which is a state-of-the-art baseline and which is the only existing system for our task. We further compare different variants of our systems to assess the importance of end-to-end vs. 2-stage training.

We make the following observations. (1) The performance in entity detection is generally better for arXivdocs-target (scientific articles) than for E-Periodica (magazines). This demonstrates that our new dataset is a challenging, real-world setting for evaluation. In particular, the performance difference can be explained by that the format of magazines is characterized by a fairly large variability compared to scientific articles. (2) Our DSG with end-to-end training consistently performs best. In particular, it outperforms the state-of-the-art DocParser from (Rausch et al., 2021). For example, for an IoU threshold of 0.5, the mAP of our end-to-end DSG is 1.91 percentage points better than DocParser for arXivdocs-target, and it is 7.03 percentage points better for E-Periodica. This translates into a relative improvement of 2.46% and 12.71%, respectively. (3) The larger relative performance gain for E-Periodica than for arXivdocs-target is likely due to the fact that our system can directly leverage annotated relations and learn against them, whereas DocParser is limited to simple heuristics. (4) Our DSG using end-to-end training outperforms the 2-stage training approach. Hence, one of the reasons for the strong performance of our approach is the joint learning procedure, in which components C2 and C3 allow for system-wide parameter updates. In sum, the results demonstrate the effectiveness of our DSG.

Table 4.7 further provides a breakdown by different semantic categories. Evidently, our DSG is consistently better for the vast majority of semantic categories. For example, for an IoU threshold of 0.5, it achieves an improvement by 9.34 percentage points for the ARTICLE category. Entities of this category are important during relation classification, because they reflect the
high-level segmentation of document pages and, thus, are used in a large number of hierarchical relations. Evidently, the end-to-end training objective of DSG that incorporates relation-level losses provides the system with useful supervision signals for this category. DocParser scores slightly better than our system for the header category at an IoU threshold of 0.5. We hypothesize that this could be due to the fact that header entities only account for 1.38% of all entities and are less relevant for parsing the document structure, since they are often not part of the reading order in magazine articles. As such, DSG is less incentivized to optimize for this entity category through its end-to-end training objective.

<table>
<thead>
<tr>
<th>System</th>
<th>Variant</th>
<th>IoU=0.5</th>
<th>IoU=0.75</th>
</tr>
</thead>
<tbody>
<tr>
<td>DocParser (Rausch et al., 2021)</td>
<td>–</td>
<td>77.70</td>
<td>58.62</td>
</tr>
<tr>
<td>DSG 2-stage (ours)</td>
<td>w/ C2 frozen</td>
<td>71.03</td>
<td>50.48</td>
</tr>
<tr>
<td>DSG 2-stage (ours)</td>
<td>w/ C2 unfrozen</td>
<td>77.48</td>
<td>57.40</td>
</tr>
<tr>
<td>DSG end-to-end (ours)</td>
<td>w/o postprocessing</td>
<td><strong>79.61</strong></td>
<td><strong>58.58</strong></td>
</tr>
<tr>
<td>DSG end-to-end (ours)</td>
<td>w/ postprocessing</td>
<td><strong>79.61</strong></td>
<td><strong>58.58</strong></td>
</tr>
</tbody>
</table>

Table 4.5: Performance (mAP) of entity detection for dataset arXivdocs-target.

<table>
<thead>
<tr>
<th>System</th>
<th>Variant</th>
<th>IoU=0.5</th>
<th>IoU=0.75</th>
</tr>
</thead>
<tbody>
<tr>
<td>DocParser</td>
<td>–</td>
<td>55.32</td>
<td>35.33</td>
</tr>
<tr>
<td>DSG 2-stage (ours)</td>
<td>w/ C2 frozen</td>
<td>51.51</td>
<td>10.84</td>
</tr>
<tr>
<td>DSG 2stage (ours)</td>
<td>w/ C2 unfrozen</td>
<td>54.41</td>
<td>36.22</td>
</tr>
<tr>
<td>DSG end-to-end (ours)</td>
<td>w/o postprocessing</td>
<td><strong>62.35</strong></td>
<td><strong>41.26</strong></td>
</tr>
<tr>
<td>DSG end-to-end (ours)</td>
<td>w/ postprocessing</td>
<td>62.18</td>
<td><strong>40.90</strong></td>
</tr>
</tbody>
</table>

Table 4.6: Performance of entity detection for dataset E-Periodica.

**Structure generation:** We now evaluate the accuracy with which the hierarchical relations are correctly generated. For this, we again report the performance for both datasets, namely, arXivdocs-target (Table 4.8) and E-Periodica (Table 4.9).

We make the following observations. (1) We again measure an overall better performance for arXivdocs-target than for E-Periodica. This is expected due to the complex format of magazine articles. (2) We find that our DSG performs best. In particular, it outperforms the state-of-the-art...
DocParser (Rausch et al., 2021) from the literature by a clear margin. Our system improves over the F1 from DocParser by 7.63% (arXivdocs-target) and 183.44% (E-Periodica). (3) We again find a larger improvements for E-Periodica than for arXivdocs-target. This can be explained

Table 4.7: Performance of entity detection per category for dataset E-Periodica. Reported: average precision (AP) per semantic category. Compared: DocParser and DSG end-to-end.

<table>
<thead>
<tr>
<th>Semantic category</th>
<th>DocParser (Rausch et al., 2021)</th>
<th>DSG (ours)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IoU=0.5</td>
<td>IoU=0.75</td>
</tr>
<tr>
<td>mAP</td>
<td>55.32</td>
<td>35.33</td>
</tr>
<tr>
<td>ARTICLE</td>
<td>66.51</td>
<td>48.69</td>
</tr>
<tr>
<td>AUTHOR</td>
<td>41.32</td>
<td>16.92</td>
</tr>
<tr>
<td>BACKGR._FIG.</td>
<td>53.69</td>
<td>34.66</td>
</tr>
<tr>
<td>COLUMN</td>
<td>45.36</td>
<td>23.44</td>
</tr>
<tr>
<td>TEXTBLOCK</td>
<td>78.03</td>
<td>66.86</td>
</tr>
<tr>
<td>DOC._ROOT</td>
<td>99.01</td>
<td>99.01</td>
</tr>
<tr>
<td>FIGURE</td>
<td>42.10</td>
<td>19.08</td>
</tr>
<tr>
<td>FIG._CAPTION</td>
<td>32.58</td>
<td>23.15</td>
</tr>
<tr>
<td>FIG._GRAPHIC</td>
<td>66.29</td>
<td>50.10</td>
</tr>
<tr>
<td>FOOTER</td>
<td>42.59</td>
<td>8.51</td>
</tr>
<tr>
<td>FOOTNOTE</td>
<td>57.97</td>
<td>45.83</td>
</tr>
<tr>
<td>HEADER</td>
<td><strong>44.83</strong></td>
<td><strong>15.87</strong></td>
</tr>
<tr>
<td>HEADING</td>
<td>57.61</td>
<td>26.07</td>
</tr>
<tr>
<td>ITEM</td>
<td>46.76</td>
<td>30.62</td>
</tr>
<tr>
<td>ITEMIZE</td>
<td>53.72</td>
<td>40.87</td>
</tr>
<tr>
<td>META</td>
<td>83.97</td>
<td>83.97</td>
</tr>
<tr>
<td>ORDEREDGROUP</td>
<td>65.51</td>
<td>42.02</td>
</tr>
<tr>
<td>PAGE_NR.</td>
<td>62.93</td>
<td>1.80</td>
</tr>
<tr>
<td>ROW</td>
<td>49.75</td>
<td>17.00</td>
</tr>
<tr>
<td>TABLE</td>
<td>52.87</td>
<td>39.38</td>
</tr>
<tr>
<td>TABLE_OF_CONT.</td>
<td><strong>51.90</strong></td>
<td><strong>21.99</strong></td>
</tr>
<tr>
<td>TABULAR</td>
<td>52.58</td>
<td>37.84</td>
</tr>
<tr>
<td>UNORDERED_GROUP</td>
<td>24.52</td>
<td>18.99</td>
</tr>
</tbody>
</table>
4.7. Results

by that our system can directly leverage annotated relations and learn against them, whereas DocParser is limited to simple heuristics. (4) Our DSG benefits from end-to-end training. As can be seen in our ablation studies, end-to-end training outperforms 2-stage training. (5) We observe a slight drop in F1 scores after applying postprocessing the hierarchical structures $H$ produced by DSG. A reason for this lies in our strict evaluation procedure, paired with the motivation of producing valid tree structures as a result of our postprocessing. To illustrate this, let us consider an entity that is missed by our system. If this node would be normally be positioned as an intermediate node in the document, our postprocessing could connect its successor and predecessor entities to form a valid document structure. This would, however, have a negative effect on overall performance, but facilitates our aim of generating valid tree structures.

The above evaluation has also an important implication. DocParser builds on heuristics that were specifically tailored to scientific articles in the arXivdocs-target dataset. For this reason, DocParser is not directly effective for other datasets such as E-Periodica without manual re-engineering.

We remind that we enforce a strict evaluation in which the complete tuple including both entities must be correct. Hence, our structure parsing task relies on the accurate identification of every entity and the relation type in a given triplet. Because of this, high F1 scores require a high detection accuracy in entity recognition. Nevertheless, the performance of our system is highly effective in practice where the aim is to recover the overall document structure. We demonstrate this through a qualitative assessment in the following section.

<table>
<thead>
<tr>
<th>System</th>
<th>Variant</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>DocParser (Rausch et al., 2021)</td>
<td>–</td>
<td>0.6646</td>
<td>0.7687</td>
<td>0.7054</td>
</tr>
<tr>
<td>DSG 2-stage (ours)</td>
<td>w/ C2 frozen</td>
<td>0.7689</td>
<td>0.7042</td>
<td>0.7223</td>
</tr>
<tr>
<td>DSG 2-stage (ours)</td>
<td>w/ C2 unfrozen</td>
<td>0.7378</td>
<td>0.7560</td>
<td>0.7378</td>
</tr>
<tr>
<td>DSG end-to-end (ours)</td>
<td>w/o postprocessing</td>
<td>0.7709</td>
<td>0.7649</td>
<td>0.7592</td>
</tr>
<tr>
<td>DSG end-to-end (ours)</td>
<td>w/ postprocessing</td>
<td>0.6959</td>
<td>0.7590</td>
<td>0.7185</td>
</tr>
</tbody>
</table>

Table 4.8: Performance of structure parsing for dataset arXivdocs-target.
Table 4.9: Performance of structure parsing for dataset E-Periodica.

<table>
<thead>
<tr>
<th>System</th>
<th>Variant</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>DocParser (Rausch et al., 2021)</td>
<td>–</td>
<td>0.1725</td>
<td>0.2319</td>
<td>0.1884</td>
</tr>
<tr>
<td>DSG 2-stage (ours)</td>
<td>w/ C2 frozen</td>
<td>0.3901</td>
<td>0.5083</td>
<td>0.4232</td>
</tr>
<tr>
<td>DSG 2-stage (ours)</td>
<td>w/ C2 unfrozen</td>
<td>0.4589</td>
<td>0.5276</td>
<td>0.4740</td>
</tr>
<tr>
<td>DSG end-to-end (ours)</td>
<td>w/o postprocessing</td>
<td>0.5545</td>
<td><strong>0.5528</strong></td>
<td><strong>0.5340</strong></td>
</tr>
<tr>
<td>DSG end-to-end (ours)</td>
<td>w/ postprocessing</td>
<td><strong>0.5701</strong></td>
<td>0.5197</td>
<td><strong>0.5308</strong></td>
</tr>
</tbody>
</table>

4.7.2 Qualitative Evaluation

We performed a qualitative evaluation to demonstrate the effectiveness of DSG in practice. For this, we randomly sampled a small subset of documents and then compare our system against both DocParser and commercial OCR systems.

**Procedure:** We use two state-of-the-art, commercial OCR systems: ABBYY (ABBYY, 2023) and Adobe Acrobat (Adobe, 2023). These OCR systems are able to parse content in different ordering (e.g., top-down and left-right) but without hierarchical information. Further, the output of these OCR systems is limited to a small set of semantic entities and does not focus on preserving the hierarchical document structure (i.e., the nesting). For comparison, we perform a page recognition on the evaluation documents and export the parsed pages to HTML files. We then map the generated results and HTML tags onto the closest matching semantic entities used by DSG for comparison. For instance, text regions that are wrapped by a heading tag in HTML are shown as a header bounding box in our qualitative evaluation, while text regions enclosed by a aside entity are assigned the category unordered group. We manually specify the input language (e.g., English, German) before running the tool if the tool provides such an option.

**Results:** Figure 4.6 shows examples. We shortened repeating sequences of entities for better readability. In rare cases, this can occlude potentially missing relations of type followed by. The unedited input images are shown in Figures 4.3a, 4.4a and 4.4b. Overall, we find that, even for F1 scores in the order of ~0.5, the final document structure is typically very accurate. While even minor discrepancies or ambiguities between the predicted entities and the ground-truth may lead to notable drops in F1 scores, their overall similarity is still large. Additional examples are in the Appendix.
4.7. Results

(a) Ground-truth.  (b) ABBYY (ABBYY, 2023).  (c) Acrobat (Adobe, 2023).  (d) DocParser (Rausch et al., 2021).  (e) DSG (ours).


Figure 4.6: Qualitative evaluation comparing the parsed hierarchical document structure by different systems. The document is characterized by complex, hierarchical structure. Top: entity recognition; bottom: hierarchical structure.
4.8 Querying

4.8.1 hOCR Querying

We further support direct querying of hOCR files for downstream tasks as follows. To this end, we introduce an extended DSG syntax for queries using XPath (XML Path Language).

We provide example queries to underline the functionality:

- Using the DSG document structure in the enriched hOCR files allows more complex queries such as 
  
  ```
  //div[@dsg_cat="orderedgroup"]/*/div[@dsg_cat="heading"]/span[@text="results"]/...
  ```

  This query returns all heading entities that contain the word “results” and are a child of an ordered group entity.

- We enable queries on sequential order, e.g., written as 
  
  ```
  followedby(//div[@dsg_cat="heading"], //div[@dsg_cat="textblock"]) 
  ```

  This query returns all text block entities that follow a heading.

  The `followedby(...)` method takes two lists of DSG XML nodes or two xPaths as input and returns all nodes from the second list that follow a node from the first list.

4.8.2 Examples

We demonstrate an exemplary query on real data (Fig. 4.6j) as follows:

```python
>>row_child_of_tabular_and_containing_diplome=root_hocr.xpath('//div[@dsg_class="tabular"]/*/div[@dsg_class="row"]/span[text()="Diplome"]/..')
>>print(entity_child_texts(row_child_of_tabular_and_containing_diplome))
['Institutionen', 'Kurse', 'Diplome', 'XII']
>>headings=root_hocr.xpath('//div[@dsg_class="heading"]')
>>print_heading_text(headings[:2])
['Das Wallis im Profil', 'Biographie', '-', 'Bibliographie', 'Maurice', 'Chappaz']
>>textblock_after_biblio=followedby('//div[@dsg_cat="heading"]/span[text()="Biographie"]/..', '//div[@dsg_cat="contentblock"]', root_hocr)
>>print(entity_child_texts(textblock_after_biblio)[:5])
['Geboren', 'am', '21.12.1916', 'in', 'Martigny']
```
4.9 Discussion

**Novel system:** Our system is relevant for several downstream tasks for which document renderings (e.g., PDF files, scans) must be mapped onto a parseable format. Examples are (Che et al., 2006; Cafarella et al., 2008; Wilkinson, 1994; Li and Moon, 2001; Manabe and Tajima, 2015; Wu et al., 2018). Recent works (e.g., (Appalaraju et al., 2021; Huang et al., 2022)) have introduced transformer-based systems for large-scale pre-training on document data but for other tasks such as entity detection and thus without extracting hierarchical structures. Hence, our system is orthogonal to such works and makes an important, non-trivial contribution. Importantly, the main advantage of our DSG is that it can generate complete hierarchical document structures through end-to-end training.

**Comparison to OCR systems:** Prior research (e.g., (Li et al., 2020a)) has repeatedly demonstrated the challenges in existing OCR systems. OCR systems are typically not designed for generating hierarchical document structures but primarily for inferring textual contents from document renderings. As a result, OCR systems generally struggle with recognizing fine-grained structures such as subfigures and their ordering (see our qualitative analysis above). Our system alleviates these challenges and is thus specifically designed to accurately generate hierarchical document structures with a high granularity to enable downstream tasks. To this end, we opted for relatively strict performance metrics to ensure that also fine-grained structures are recognized correctly.

**Practical strengths:** A key strength of our system is that it is end-to-end trainable. This allows our system to take full advantage of existing training data, including information on the hierarchical relations that captures the sequence and nested structure within documents. In contrast, prior systems (Rausch et al., 2021) are not end-to-end trainable but infer relations through heuristics, thereby essentially ignoring the corresponding information in the training data. As a result, our system reduces the cost for annotating hierarchical document structures by a significant extent. In sum, our system fulfills a key demand in practice where the generation of document structures is often subject to scarce data and where systems should be customizable in a flexible manner.

**Novel dataset:** We contribute a novel, large-scale dataset based on magazines for generating hierarchical document structures. In particular, our dataset provides a challenging real-world setting for evaluation due to the large heterogeneity in the layout of magazines. Key to our dataset is its large granularity of the annotations in terms of both fine-grained entities and the
relations between them. This is different from other datasets, which are typically coarse (Zhong et al., 2019) and without hierarchical information (Li et al., 2020c).

**Future work:** Our experiments also show that the entity detection from prior research can be improved through our end-to-end training procedure. For example, let us compare the results of DocParser against DSG, both of which are based on the same entity detection, thus confirming the effectiveness of our end-to-end training. Hence, there is a large potential to improve other tasks around entity detection in documents through end-to-end training as part of future research.

## 4.10 Summary

In this chapter, we introduced *Document Structure Generator* (DSG), a novel system for parsing hierarchical document structures that is end-to-end trainable. We show that our system outperforms state-of-the-art systems. By being end-to-end trainable, our DSG is of direct value in practice in that it can be adapted to new documents in a straightforward manner without the need for manual re-engineering. Furthermore, our DSG generate output in the hOCR format and thus allows for seamless integration into existing document storage and processing workflows.
5.1 Introduction

The systems for hierarchical structure parsing presented in Chapters 3 and 4 are primarily designed for parsing hierarchical document structures. For retrieving full marked-up text documents from renderings, such systems need to be further integrated into document optical character recognition (OCR) systems.

Document OCR, cf. Fig 1.1, is an important step in the process of digitizing documents and texts. Current state-of-the-art OCR systems typically separate into two main processing stages: First, hierarchical structures are parsed in the document. Second, systems process individual text lines or words through text recognition. In modern systems, this processing typically involves one or more trained machine learning models.

OCR subtasks have been continuously improved in recent years, achieving high performance for text recognition (Breuel, 2017; Hernandez Diaz et al., 2021) and document entity detection (Zhong et al., 2019; Appalaraju et al., 2021). However, end-to-end OCR requires the transformation of detected document entities and recognized text into structured, marked-up text, which is still a challenging problem.
Chapter 5. LayTr: Differentiable end-to-end OCR for Complex Layouts with Transformers

Current approaches to end-to-end OCR are predominantly reliant on heuristics and extensive engineering efforts for complete systems (Zhu et al., 2022) or structure parsing (Rausch et al., 2021). There are also systems that treat structure parsing as a learning problem, e.g. (Holeček et al., 2019) or (Rausch et al., 2023a), but not for the end-to-end OCR task that includes text recognition. As a result, end-to-end OCR remains a problem in practice, and systems that treat full document OCR as an end-to-end learning task are missing.

Dividing structure parsing and text recognition into separate stages means that structure parsing cannot be informed by textual content. This is particularly limiting for multi-column documents, where linguistic information can help determine whether a partially recognized line has reached a column boundary or continues (see also (Hurst and Nasukawa, 2000; Prieto et al., 2020) for the utility of linguistic information for document structure parsing). This may be addressed by having a pre-recognition entity detection stage (e.g., of words), followed by text recognition, and followed by a second structure parsing stage.

Disjoint systems also tend to be complex and difficult to engineer and optimize. First, we need to choose data structures and representations for the different recognition results. Computing the intermediate representations also requires a choice of thresholds or parameters. Furthermore, each processing stage depends on the outputs of the previous stage, requiring retraining when a previous stage or manually chosen parameters have changed.

End-to-end training of a single deep model performing structure parsing and text recognition in an integrated manner avoids all these problems: it greatly simplifies the software architecture, eliminates many manual parameter choices, and allows the simultaneous use of layout and text content information for informing structure parsing and text recognition decisions.

Objective: Our aim is to build an end-to-end OCR system that can directly and robustly map document renderings to marked-up text documents.

This kind of end-to-end training has now become possible with the introduction of transformer models (Vaswani et al., 2017; Dosovitskiy et al., 2023; Liu et al., 2019b). Transformers are deep learning models that employ a self-attention mechanism. This allows them to flexibly learn the importance of the different parts of inputs at different processing steps. Transformers have been shown to be effective for multimodal document processing and pretraining tasks relating to entity detection and classification (Xu et al., 2020; Appalaraju et al., 2021), or text recognition (Li et al., 2022b), but not for full document OCR.

Approaching OCR as a fully end-to-end trainable system means that evaluation and comparison
with traditional OCR systems are more difficult since typically OCR systems tend to be benchmarked separately on entity detection (e.g., using IOU criteria as described in 2.2.2), hierarchical structure parsing (e.g. by F1 scoring), or text recognition (using character or word error rates). In this chapter we choose an end-to-end benchmarking approach based on Levenshtein distance, Block Levenshtein distance, and n-gram multiset differences; these allow us to quantify both text recognition and structure parsing errors from end-to-end output.

5.1.1 Contributions

In this work, we introduce LayTr, an end-to-end trainable OCR system for documents. LayTr is based on transformer models. It takes images of texts with complex layouts as input and produces the corresponding machine-readable text as output. Our contributions to the fields of document analysis and OCR are as follows:

1. LayTr, a unified system for joint structure parsing and text recognition that can be easily deployed and is fully end-to-end trainable in a single stage. This makes it highly effective for application in document OCR tasks.

2. We create and provide two large-scale synthetic datasets for model training, evaluation, and comparison. In addition, we employ a range of evaluations to assess both structure parsing and text recognition performance of end-to-end systems where intermediate structure parsing results are not available.

3. We demonstrate that LayTr is capable of robustly recognizing complex document inputs with a variety of fonts, formats, and quality degradation while outperforming state-of-the-art commercial and open-source systems for end-to-end OCR.

5.1.2 Overview

We provide an overview of the related work in terms of systems and benchmarks in Section 5.2. In Section 5.3 we describe the two novel datasets. Section 5.4 contains details on the evaluated systems, training protocols, and our evaluation. We present and interpret our results in Section 5.5 and feature a discussion in Section 5.6. Lastly, a summary of the chapter is provided in Section 5.7.

---

1All datasets, system, and data processing code will be available open-source.
Chapter 5. LayTr: Differentiable end-to-end OCR for Complex Layouts with Transformers

5.2 Related work

Typical OCR systems are based on two primary processing stages: Structure parsing and text recognition (Subramani et al., 2020). As a result, there exists a variety of system and benchmarks that address different subproblems of these stages.

5.2.1 OCR system components

Text recognition via line- or word-based optical character recognition has long been studied. In the past, rule-based systems were prevalent (Mori, Shunji et al., 1999). Current OCR systems employ deep learning techniques such as convolutional neural networks (CNN) or long short-term memory (LSTM) models (Breuel, 2008; Breuel et al., 2013). These methods are typically trained in conjunction with a connectionist temporal classification (CTC) loss or transformers with recursive decoding (Hernandez Diaz et al., 2021).

Structure parsing and entity detection methods can be categorized into different subdomains. We discuss methods for the detection of coarse- and fine-scale document entities, as well as methods for classifying relations that describe the interconnection of entities, i.e. the reading order of the document. On a coarser scale, document entities such as paragraphs, tables, figures, captions, headings, titles, etc. are identified. In recent years, research has focused on the collection of large-scale datasets and the training of deep neural networks for the automatic detection of these entities (Zhong et al., 2019; Rausch et al., 2021; Li et al., 2020c). Traditionally, heuristic-based methods have been developed to perform more fine-grained detection for subsequent OCR (Shafait et al., 2008; Razak et al., 2008; Bukhari et al., 2013; Eskenazi et al., 2017). More recently, deep neural networks have been used for the detection of text lines (Renton et al., 2018; Mechi et al., 2019). Transformers have been applied to some domains of OCR, but not to end-to-end document OCR (Li et al., 2022b; Ströbel et al., 2022). Other work has explored the use of transformer decoders in conjunction with CNNs for OCR on text lines (Wick et al., 2021; Hernandez Diaz et al., 2021).

5.2.2 Datasets and benchmarks

The two processing stages are subject to ongoing research in a range of subdomains. Current research and benchmarks focus primarily on improving and evaluating the tasks related to one of the individual two stages, rather than unified end-to-end systems. As such, they are not directly
applicable to end-to-end OCR in the investigated setting, where we want to predict marked-up text outputs directly from document renderings.

**Text recognition** has been evaluated on text lines extracted from printed documents, for example, the UW3 dataset (Phillips, Ihsin Tsaiyun, 1996; Breuel et al., 2013). Here, character error rates (CER) of 0.4% were measured for (Breuel, 2017). Additionally, OCR is performed and evaluated for handwritten text datasets, with current state-of-the-art systems achieving CERs of 2.75 (Hernandez Diaz et al., 2021) and 1.9 (Bluche and Messina, 2017) on the IAM (Marti and Bunke, 2002) and RIMES (Grosicki et al., 2009) datasets, respectively. The described evaluations commonly evaluate on settings without document structure parsing beyond the processing of text lines.

**Scene Text Recognition (STR)** models aim to correctly localize and identify individual words in natural images that feature text. More than 10 frequently featured benchmark datasets (Naiemi et al., 2022) exist for STR. For the ICDAR 2013 (Karatzas et al., 2013) and ICDAR 2015 challenges (Karatzas et al., 2015), the state-of-the-art performance of 98.4 (Li et al., 2022b) and 88.7 (Lyu et al., 2022) has been achieved, as measured by word prediction accuracy (WPA). Unlike document OCR, STR only aims at localizing a set of words without ordering them.

**Document entity detection** is performed at different granularities, and a variety of evaluation metrics are used to measure performance, based on the granularity and the family of parsing methods. On a coarser scale, document entities such as tables, figures, headers, paragraphs, etc., are featured in datasets such as (Zhong et al., 2019; Clausner et al., 2019; Rausch et al., 2021; Smock et al., 2022). Further datasets for printed documents feature word-level entity annotations (Li et al., 2020c) or line-level entity annotations (Shafait, 2007; Eskenazi et al., 2017). The mean average precision (mAP) value indicates the overall average performance of the system in different categories of entities, such as tables, paragraphs, or figures. Current state-of-the-art results for entity detection are achieved by (Zhang et al., 2021) with an mAP of 95.69 on (Zhong et al., 2019) and 87.6 on (Li et al., 2020c). The task of document entity detection is, however, not directly applicable to full document OCR, where additional processing of document structure and text recognition is required.

**Hierarchical structure parsing** benchmark tasks further annotate hierarchical nesting or reading order at different levels. On a coarse scale, an F1 score of 0.615 was achieved for the task of correctly identifying the reading order and the nesting relations between coarse entities (Rausch et al., 2021). At the word level, (Zhu et al., 2022) achieves a page-level string edit distance of 0.38 on the (Clausner et al., 2019) dataset, combining entity detection (Zheng et al., 2021) with an OCR system via heuristics. These systems rely on heuristics and dedicated training of
Chapter 5. LayTr: Differentiable end-to-end OCR for Complex Layouts with Transformers

the entity detection components, as well as external OCR tools. Furthermore, training is required for the entity detection components, for which annotations are not necessarily available in end-to-end document OCR.

**Word entity detection or classification** is performed in other benchmarks, e.g. for individual words in scanned receipts (Huang et al., 2019) or in forms (Jaume et al., 2019). Current state-of-the-art results are achieved by (Li et al., 2022b) for the task of word recognition on scanned receipt data (96.58 F1), which does not incorporate full document structure. The system of (Huang et al., 2022) performs semantic word classification in form documents with an F1 score of 92.08, however, this step builds on prior parsing by an OCR system. Various methods have been proposed for line entity detection in printed documents (Eskenazi et al., 2017). In (Shafait, 2007), the performance of line detection and dewarping is measured on single-column images by proxy through the Levenshtein distance. Here, a Levenshtein distance of 0.025 is achieved by (Kil et al., 2017). These systems, however, do not feature more challenging layout problems such as multicolumn disambiguation.

**End-to-end document OCR** evaluations are, in summary, limited. While direct comparison with explicit structure parsing methods is challenging, proxy measurements for comparable methods yield Levenshtein distances of 0.025 (single-column) (Kil et al., 2017) and 0.38 (full-page layouts) (Zhu et al., 2022).

### 5.3 Datasets

In previous work, transformer-based OCR systems have been trained and evaluated on isolated text lines. In our experiments, we investigate the capability of LayTr to effectively process text with multicolumn and tabular layouts.

Our initial implementation, which is based on TrOCR, cannot accept full-page images, due to restrictions of the pre-trained models. In this work, we are therefore using artificially generated training data for multicolumn text and tables corresponding to roughly a quarter of a printed page.

#### 5.3.1 LayTr-Columns

Our aim is to build a dataset that is adequate for large-scale training and evaluation of our transformer model. To do this, we build a custom data generator. The data generator retrieves
random text sections from real-world literature and renders them into double-column formatted
text images.

We randomly sample English texts from the Gutenberg Project repository (Gutenberg, 2023). For each individual sampled book text, we extract up to 200 text sections. Each text section has a randomly chosen length of 50 to 100 words. The paragraph texts are adapted by replacing the new line characters with spaces.

For the generation of documents that use real-world typesetting, we set up an HTML-based rendering engine. For this, we generate custom HTML and CSS files to specify the content and style of each individual sample. We use this engine to generate a rendered image of double-column text for a given input string by wrapping it into an appropriate HTML format. The original input text later serves as the ground truth for training and evaluation.

To achieve a wide variety of appearances, we randomly choose fonts, alignment of the generated text samples, and spacing between the two columns. Text fonts are sampled from four different categories. We use the font categories Serif, Sans Serif, Monospace, and Cursive in our experiments. In total, our dataset includes 23 individual fonts. We randomly set the alignment of the text per sample to one of four settings. The alignment settings are justify, center, left, and right.

The number of lines per column is not manually specified. Instead, we randomly select a column width and spacing between columns. The text content is then automatically distributed by the rendering engine. The column gap is chosen randomly from a range of 2 to 30 pixels. The column width is selected to be between 30 and 60 \text{em} units. The unit \text{em} is a unit of size relative to the font size used. We use double-column layouts for the data generated in LayTr-Columns.

Using our rendering engine, we create a dataset of 873,596 rendered text samples with complex column layouts for the training LayTr. We reserve a holdout dataset that contains 490 samples for evaluation. Samples of renderings featured in our dataset are shown in Figure 5.1. We invert the black-and-white coloring of the samples according to the expected color schema of the investigated OCR system.

5.3.2 LayTr-Tables

We also contribute a second dataset, LayTr-Tables, which consists of tables with diverse layouts and fonts. We use this dataset to investigate the ability of LayTr to learn to parse even more complex layout input. While the ground truth texts in LayTr-Columns are formatted as plain text, the ground truth in LayTr-Tables contains explicit HTML notation. As such, we can investigate
the top of the head,” put in Marcy, who stood behind the real estate dealer. “What do you mean by staying around this island after I ordered you away?” went on Hiram Skeetles, after a pause. “Did you expect us to do any traveling in this storm?” asked Joel Runnell, in return. “How far do you think you could travel, Mr. Skeetles?” asked Fred. “The

(a) about wailing winds and weeping skies; for mine is not “a poet’s vision dim,” but an open-eyed, scientific sight of things as they actually are. Once I have seen them, gathered them, if then they turn to poetry, let them turn. For so does the squash turn to poetry when it is brought in from the field. It turns to pie; it turns to poetry; and it still remains squash. Good nature-literature, like all good literature, is more lived

(b) hold her head pretty high, but that day she held it drooping a little and her black eyes cast down. Ronald Fraser was very tall and fair, with blue eyes. They made as handsome a couple as I ever saw. “But old

James Gordon and Thomas and Janet didn’t much approve of him. I saw that plain enough one time I was there and he brought Margaret home from Radnor Friday night.

(c) of midges and rapidly darting first at one end then at another secure half a dozen of the tiny flies before the column was broken up; then retire to a branch and wait until it was re-formed, when it made another sudden descent on them.

I have no doubt many humming-birds suck the

(d)
5.4. Experimental details

In this section, we provide details on our system and its training procedure. Furthermore, we describe the commercial and open-source systems for end-to-end OCR that we compare LayTr with, as well as the our evaluation methods.

5.4.1 LayTr system

LayTr is based on the TrOCR architecture (Li et al., 2022b) and follows a transformer-decoder structure. It receives as input rendered document images $D_{i}, i = 1, \ldots, n$ of fixed resolution.
and outputs a sequence of word pieces. The concatenation of these words results in strings of plain text or marked-up text \( M_i, i = 1, \ldots, n \), depending on the type of learned task. Our system consists of a vision transformer encoder (Touvron et al., 2021) that is coupled with a language transformer decoder (Liu et al., 2019b). Figure 5.3 shows an overview of our architecture for the exemplary input of an image rendering of a line of text. We provide additional information on the different processing stages as follows.

**Image preprocessing:** Our source input data consists of rendered images of text documents with a resolution of 300 DPI (dots per inch). For processing with LayTr, we adapt the images to a fixed resolution of 384 × 384 pixels as in (Li et al., 2022b), resulting in a quality similar to that of fax images. First, the images are rescaled so that their longest side has a length of 384 pixels. For rescaling, we apply bilinear interpolation. Afterward, the images are zero-padded so that both sides have equal size. In contrast to the implementation of (Li et al., 2022b), we choose a ratio-preserving method for resizing. This is done to ensure a more consistent appearance of the rendered text input. By default, all systems operate on the resized input in our evaluations.

During training, we apply random augmentation to increase the variety of inputs and reduce overfitting of the model. Augmentation is performed using document-specific augmentation operations of the ocrodeg library as described in (Breuel, 2023). Our augmentation consists of several steps. First, the images are zero-padded by 5% of the longest edge size to account for subsequent image distortions. We randomly rotate the input by up to 2°. With a probability of 0.5, we randomly distort the images using the `distort_with_noise()` method. The random
5.4. Experimental details

distortion is based on a mesh grid that is transformed by random Gaussian noise. Noise is specified by a random sigma value between 2 and 20, and its delta value is limited to 5. With a probability of 0.5, we subsequently perform a set of document-specific augmentation steps through the printlike_multiscale() function of ocrodeg. Here, the blobs in the image are randomly blurred. In addition, paper- and ink-like noise is randomly added to inputs.

**Target text processing:** Text inputs are encoded using Byte Pair Encoding (BPE) to generate word pieces corresponding to the outputs of LayTr. Before BPE, a pre-tokenization step is performed, analogously to GPT-2 (Radford et al., 2019) and RoBERTa (Liu et al., 2019b). An "[EOS]" token is appended to the encoded texts during training to signal the end of the input sequence. Our encoder uses a dictionary of size 50,265 that is based on GPT-2.

**Encoder:** The transformer encoder utilized in LayTr expects a sequence of input features that correspond to rectangular patches of the input image. For this, the preprocessed document images are split into patches of 16x16 pixels, flattened, and linearly projected to patch embeddings of dimension 768. Position embeddings, which correspond to the absolute position of the individual patches in the input image, are added to the patch embeddings to form the encoder inputs. Our transformer encoder has a depth of 12, a hidden size of 768, and uses 12 attention heads.

**Decoder:** The transformer decoder has a depth of 12, a hidden size of 1024, and uses 16 attention heads. Attention masking is used in the decoder during training to prevent it from using the information of future word pieces in the sequence. This is to ensure that the decoder does not have access to more information during training than it would during inference.

5.4.2 Training procedure

Our experiments are executed with the Fairseq sequence modeling framework (Ott et al., 2019). The transformer encoder is initialized with the stage one TrOCR-base weights. The transformer decoder is initialized with pretrained RoBERTa weights. We use an initial learning rate of $2.0 \cdot 10^{-9}$ and a batch size of 16. We use the Adam optimizer (Kingma and Ba, 2015) in our experiments. We employ weight decay throughout training, based on the inverse square root of the update number. During training, images are randomly sampled from the training dataset. We define a training epoch as the training of our model on 250,000 sampled examples. In our experiments, we performed training on 8 GPUs for a total of 31 and 77 epochs on LayTr-Tables and LayTr-Columns, respectively. To account for the greater number of tokens in LayTr-Tables, we increase the maximum number of target tokens in our system to 2048 in our experiments on LayTr-Tables. We base additional hyperparameter choices on (Li et al., 2022b) for our ex-
Chapter 5. LayTr: Differentiable end-to-end OCR for Complex Layouts with Transformers

5.4.3 Compared OCR systems

We compare our model with state-of-the-art systems for end-to-end document OCR. Specifically, we choose the commercial software "ABBYY FineReader" (ABBYY, 2023) and the open-source system "Tesseract" (Smith, 2007) as representative end-to-end OCR systems. In this study, we only consider end-to-end OCR systems and therefore no current systems for exclusive entity detection that would require further heuristics. Furthermore, we note that as part of our end-to-end evaluation, no intermediate bounding box labels for the training of entity detectors are given.

Tesseract is configured to use default options for text recognition and structure parsing. As such, Tesseract performs fully automatic structure parsing.

For experiments on LayTr-Columns, we extract plain text-formatted content for ABBYY. The predicted texts from ABBYY and Tesseract are cleaned by removing the preceding and following whitespaces and converting tab characters into regular spaces.

For experiments on LayTr-Tables, a range of inputs would not be categorized as tabular data by ABBYY. To allow for more accurate OCR output, we correct all evaluation samples by manually annotating the image content as a tabular region. Following the correction, we perform the recognition step. We then extract HTML-formatted pages for the evaluation samples. We then manually process the HTML code so that it matches the composition and format of the ground truth by adding or removing potentially missing HTML tags and whitespaces.

In our experiments, we retrieve plain text-formatted outputs from Tesseract. To allow for a comparison of all three systems on LayTr-Tables, we create a secondary representation. Here, the predictions of all systems are processed so that the resulting string contains all row-ordered, recognized words in a sequential, space-separated format.

We perform a set of post-processing steps to make the predictions of the different systems and the ground truth texts more consistent. Post-processing is applied to all generated texts and replaces Unicode variants of certain characters with a uniform character. Specifically, we unify all dashes and single and double quotation marks. LayTr produces outputs that contain only
single quotation marks. To make the outputs more comparable, we apply a simple heuristic to the LayTr outputs. Here, any occurrence of two subsequent single quotation marks is replaced by the uniform character for the double quotation mark.

5.4.4 Evaluation

Quantitative Evaluation: Systems are evaluated by comparing their OCR outputs with the ground truth using the Levenshtein distance, Block Levenshtein distance, as well as the multiset difference of n-grams. The evaluation of existing systems for structure parsing is based on entity bounding boxes. This is not possible here since our end-to-end OCR setting does not incorporate bounding boxes. The Block Levenshtein distance, as well as the multiset difference of n-grams are alternatives that let us evaluate text recognition accuracy and structure parsing quality based only on the predicted text.

The two distance metrics used are the normalized Levenshtein distance $D_L$ and the normalized Block Levenshtein distance $D_{BL}$. For the sake of brevity, we omit the use of the term normalized, when referring to the two metrics. The default Levenshtein distance utilizes character-based insertion, replacement, and deletion operations. The Levenshtein distance between a predicted string $S$ and ground truth string $G$ can be computed via Algorithm 1.

**Algorithm 1** Calculation of Levenshtein Distance between strings $S$ and $G$.

1: function Levenshtein(String $S$, String $G$)
2:     if length($S$) == 0 then
3:         return length($G$)
4:     else if length($G$) == 0 then
5:         return length($S$)
6:     else if $S[0] == G[0]$ then
7:         return Levenshtein($S[1:]$, $G[1:]$)
8:     else
9:         return 1 + min(Levenshtein($S[1:]$, $G[1:]$), Levenshtein($P[1:]$, $G$), Levenshtein($P$, $G[1:]$))

In addition to character-based operations, the calculation of the Block Levenshtein distance allows for block-level insertion, replacement, and deletion. That is, if a block of characters exists in both strings, the basic operations can be performed for that block at the same cost as for a single character. This characteristic can be used to investigate whether the OCR error of a
system is more due to errors in the structure parsing stage. For instance, two columns could be falsely identified as one unified column. In this case, $D_{BL}$ would be significantly lower than the Levenshtein distance if most of the individual words were still correctly recognized.

We represent predictions and ground truth in two formats. First, we use the original texts. Second, we encode every unique word with a single unique character. This allows distance measurement for our systems in a setting where multiple character errors for a single word are less strongly penalized.

We also perform evaluations based on the n-grams of the ground truth and different predictions. N-grams are formed at the character level. We then compute the asymmetric multiset difference between the ground truth and the predictions of the different systems. The multiset difference is normalized through the division with the length of the ground truth set. Similarly to $D_{BL}$, multiset differences for small n-grams correspond to character-level error rates, while multiset differences for large n-grams correspond to structure parsing errors.

**Qualitative evaluation:** To qualitatively compare the generated OCR outputs with the ground truth, we present examples of input images and the corresponding texts. For an easier assessment of the accuracy of both text recognition and structure parsing, we tokenize all texts with the NLTK library, using the English language setting. Afterward, every word in the ground-truth text is assigned a unique background color, resulting in a color gradient spanning from the first to the last word (e.g. blue highlighting for the first word and progressing to red for the last word in the text). Based on this coloring schema, we compare the predicted texts with the ground truth text. If a predicted token can be matched with the ground truth, it is assigned the background coloring of the respective ground truth token. As such, the qualitative evaluation has two characteristics: First, for every token in the predicted text without background coloring, no matching ground-truth token could be found. Second, if the color gradient in the predicted text does not match the sequential order of the ground truth, structure parsing errors likely occurred during parsing (e.g. appearance of red-highlighted tokens at the beginning of the predicted text).

### 5.5 Results

We evaluate LayTr on the evaluation sets of LayTr-Columns and LayTr-Tables. We compare our results with state-of-the-art commercial and open-source software for end-to-end document OCR.
## 5.5. Results

### 5.5.1 Quantitative evaluation

<table>
<thead>
<tr>
<th>LayTr-Columns</th>
<th>LayTr-Tables</th>
</tr>
</thead>
<tbody>
<tr>
<td>char.</td>
<td>word</td>
</tr>
<tr>
<td>$D_L$</td>
<td>$D_{BL}$</td>
</tr>
<tr>
<td>LayTr</td>
<td>0.030</td>
</tr>
<tr>
<td>ABBYY</td>
<td>0.341</td>
</tr>
<tr>
<td>Tesseract</td>
<td>0.633</td>
</tr>
<tr>
<td>HTML</td>
<td>HTML-stripped</td>
</tr>
<tr>
<td>char.</td>
<td>word</td>
</tr>
<tr>
<td>$D_L$</td>
<td>$D_{BL}$</td>
</tr>
<tr>
<td>LayTr</td>
<td>0.009</td>
</tr>
<tr>
<td>ABBYY</td>
<td>0.147</td>
</tr>
<tr>
<td>Tesseract</td>
<td>0.164</td>
</tr>
</tbody>
</table>

Table 5.1: Performance of the investigated systems for different evaluation settings and datasets for Levenshtein (L) and Block Levenshtein (BL) metrics. We observe that LayTr outperforms the state-of-the-art systems across all datasets and evaluation settings. The distance $D_{BL}$ largely has the same value as the $D_L$ for LayTr, while it decreases for ABBYY and Tesseract. This indicates that more structure parsing errors occur for ABBYY and Tesseract.

Table 5.1 shows the results for all systems and datasets. We observe that LayTr outperforms both reference state-of-the-art systems across all datasets and metrics. Visualizations for a more detailed interpretation are shown in Figure 5.4 for LayTr-Columns and Figure 5.5 for LayTr-Tables.

**LayTr-Columns:** We observe in Table 5.1 and Figure 5.4a that LayTr achieves the lowest Levenshtein and Block Levenshtein distances across the three systems for both datasets. The distances $D_L$ and $D_{BL}$ have very similar values for LayTr. On the contrary, a high relative decrease can be observed for $D_{BL}$ for both ABBYY and Tesseract. The Block Levenshtein distance is more lenient for systems that did not accurately parse structure of the input. This indicates that a significant portion of OCR errors are caused by structure parsing for both ABBYY and Tesseract.

We measure the highest relative increase in $D_L$ and $D_{BL}$ between the character-level and word-level measurements on LayTr-Columns for LayTr (see Table 5.1). This suggests that badly recognized words contain more wrong characters, in relation to the other two systems. This could be caused by the pretrained transformer decoder that is used in LayTr. As such, the generated output generally contains valid words. If a word is falsely recognized, it is commonly replaced by a different word that is semantically or syntactically fitting at the given position. On the contrary, state-of-the-art systems tend to make more errors for individual challenging characters.

Figure 5.4c shows the histogram of the Levenshtein distances for LayTr and the reference OCR
systems on LayTr-columns. We observe that the Levenshtein distances for LayTr are restricted to distances up to 0.15. The commercial system has errors in all distance brackets up to 0.85. Additionally, the ABBYY results show elevated counts for distances up to 0.15, as well as distances higher than 0.8.

Inspection of samples with high DL on LayTr-Columns shows that errors often occur due to two causes. First, structure parsing errors lead to the double-column format being interpreted as a single column. Second, the resolution of 384 × 384 leads to the input images being challenging for the state-of-the-art OCR systems, particularly when the texts feature a cursive font.

We feature an additional evaluation in which ABBYY receives input with significantly higher resolution. The resulting histogram is shown in Figure 5.4d. In this setting, LayTr still outperforms the commercial OCR system in all distance buckets, including the number of samples with a distance up to 0.05. The commercial system achieves a low DL for most cases but has a very high distance for a subset of the evaluation samples. This corresponds to an elevation in the histogram for character-level Levenshtein distances around 0.65. Inspection of the affected samples reveals that these high errors are still associated with structure parsing errors.

Figure 5.4b shows the multiset differences between the n-grams of the system predictions and the ground truth. In contrast to LayTr, the multiset differences ABBYY and Tesseract increase steeply for n-gram numbers lower than 30. This can be attributed to both a higher text recognition error rate and structure parsing errors. Structure parsing errors would cause longer substrings to falsely include characters from both columns. Due to such errors, even fewer long substrings in the prediction set could still match the ground truth set.
5.5. Results

(a) Levenshtein and Block Levenshtein distances for the three evaluated systems on LayTr-Columns. LayTr outperforms both comparison systems significantly. The Block Levenshtein distance metric has a lower penalty for structure parsing errors. It remains near constant for LayTr but decreases for ABBYY and Tesseract. This indicates that high structure parsing errors occurred for the comparison systems.

(b) Normalized n-gram multiset differences for all systems on LayTr-Columns. N-grams are formed on a character level. Higher N-gram numbers are less forgiving for predictions with structure parsing errors. The steeper incline for ABBYY and Tesseract up to N-grams to a number of around 30 indicates a higher occurrence of structure parsing errors in the two systems.

(c) Histogram of Levenshtein distances for LayTr and ABBYY on LayTr-Columns. For the majority of test samples, the Levenshtein distance for LayTr is 0.10 or lower. For ABBYY we measure distances in all distance brackets up to 0.90, indicating an overall lower OCR performance.

(d) Histogram of Levenshtein distances for LayTr and ABBYY. ABBYY images use higher resolution images (300 DPI). Even for high-resolution inputs, LayTr achieves superior performance. At the same time, the commercial system makes structure parsing errors that lead to a peak at around 0.65.

Figure 5.4: Evaluations for the systems on the LayTr-Columns dataset.
Chapter 5. LayTr: Differentiable end-to-end OCR for Complex Layouts with Transformers

(a) Histogram of Levenshtein distances for LayTr and ABBYY on LayTr-Tables. The majority of samples have a Levenshtein distance under 0.05 for LayTr. Meanwhile, most Levenshtein distances are spread across buckets of up to 0.35, indicating a less robust OCR performance.

(b) Normalized N-gram multiset differences for the HTML-stripped outputs of all systems on LayTr-Tables. The curves suggest a better structure parsing performance for LayTr (compare Figure 5.4b).

Figure 5.5: Evaluations for the systems on the LayTr-Tables dataset.

**LayTr-Tables:** As with LayTr-Columns, LayTr achieves the best result for both the Levenshtein and the Block Levenshtein distances in all evaluation configurations. Figure 5.5a shows that the Levenshtein distances for LayTr are restricted to distances up to 0.10, while the distances are spread mainly up to 0.4 for the commercial system. We observe a better overall performance for ABBYY on LayTr-Tables. We attribute this to two main factors: First, due to the manual tagging of every input document as a table, there are fewer catastrophic structure parsing errors. Second, most tables contain fewer total words and are more compact compared to LayTr-Columns. Because of this, the inputs tend to have a higher relative resolution. As a consequence, texts are less degraded, and more challenging fonts are recognized more robustly.

We evaluate the normalized n-gram multiset differences for the HTML-stripped outputs of all systems at the character level in Figure 5.5b. The multiset difference values for Tesseract and ABBYY increase significantly more steeply than for LayTr, yet performance is better than for the LayTr-Columns dataset.

We note that the lower resolution of the samples can deteriorate the separation lines in the tables,
making structure parsing more challenging for the systems.

### 5.5.2 Qualitative evaluation

**LayTr-Columns:** We feature example outputs for LayTr-Columns for the state-of-the-art commercial system and LayTr in Figure 5.6. We only observe distances over 0.5 for LayTr for a few inputs. These inputs tend to be from sections of books in the Gutenberg library that do not correspond to the ordinary English language and feature a very high amount of special formatting and characters. We observe that for ABBYY structure parsing errors can occur for inputs with a variety of layout parameters. For instance, structure parsing for the commercial system appears to be less robust to input with cursive fonts or right-aligned texts. We also observe failures with inputs with very narrow column gaps; the commercial software expectedly does not recognize the double column format. LayTr is still able to correctly transcribe the text in these cases.

**LayTr-Tables:** For a qualitative evaluation of LayTr-Tables, we show sample image and predictions in Figure 5.7. We observe that LayTr and ABBYY can both recognize the general structure of the table. The appearance of longer words in Figure 5.7b leads ABBYY to classify several cells as multicolumn formatted, while LayTr can correctly identify all cells. More generally, we see that LayTr makes fewer token-level OCR errors.

### 5.6 Discussion

#### 5.6.1 Performance

We observe that part of the failure cases for state-of-the-art systems can be attributed to higher OCR errors for cursive font styles. However, a comparison of the systems where both ABBYY and Tesseract receive high-resolution images as input shows that LayTr still significantly outperforms the structure parsing of the other systems.

We note that further performance improvements for LayTr could be made by building on the larger models described in (Li et al., 2022b). Due to computational constraints, we used the base-sized model in our experiments. Additional hyperparameter tuning and longer training times could further increase system performance.
they saw to stand in defense of good mon and of the Senio. But did I observe the sense of the Senators themselves? I suppose that remembered how thin being present always direct me when I seem about to say or do anything. Thus remembered I say, when at Venice the King, being desirous of a common overthrow, endeavored to lay the treason, whereof only the culprit was accused, upon the whole order. We observe that a higher fraction of words correctly matches the ground truth for LayTr. Additionally, the background colors are intermixed for ABBYY, indicating that the two columns were not parsed in the correct column-by-column reading order.

Figure 5.6: Comparison of LayTr and ABBYY on two samples of LayTr-Columns (left and right). The background colors are based on the ground truth text and order. We observe that a higher fraction of words correctly matches the ground truth for LayTr. Additionally, the background colors are intermixed for ABBYY, indicating that the two columns were not parsed in the correct column-by-column reading order.
5.6. Discussion

Figure 5.7: Comparison of LayTr and ABBYY predictions on two samples of LayTr-Tables (left and right). We observe a higher word-wise OCR accuracy for LayTr. Additionally, the low spacings for some words in the input table image lead ABBYY to classify the respective table cells as double-column cells.
LayTr-Tables uses randomly chosen English words for each table cell. As such, the language model component of LayTr may perform better on real-world data, where table contents are typically more semantically related and contain more specific words.

5.6.2 Datasets

In this study, we demonstrate and assess the feasibility of LayTr for joint structure parsing and OCR. Due to the typically disjoint treatment of the two stages, new research commonly contributes to one of the two tasks and evaluates performance on the explicit results, e.g. entity detection or structure parsing accuracy. In contrast, LayTr parses the structure implicitly and does not require or produce intermediate outputs such as entity bounding boxes. For this reason, our adjusted evaluation metrics are required for direct comparison of end-to-end OCR that incorporates structure parsing quality measures.

As discussed in Section 5.3, a wide variety of methods, datasets, and benchmark protocols have been proposed for the domains of text recognition and document structure parsing. Direct comparison is difficult, but the two closest matching benchmarks measure a Levenshtein distance of 0.025 (single-column inputs) to 0.38 (complex full-page documents). LayTr achieves $D_L$ of 0.030 on complex inputs with lower resolution. This shows that LayTr has the potential to outperform existing methods for joint full-page OCR, when adapted to larger input sizes.

Due to the input size restrictions of the initial model used for this proof-of-concept study, no evaluation on full-resolution images has been carried out yet. However, the column merging problem featured in our datasets represents one of the most common and harmful kinds of structure parsing errors, which is why they were chosen as a test case. Our system can be adapted to process full-resolution inputs, which involves re-engineering of our system and more extensive pretraining on document data. After this adaptation, LayTr could be evaluated on benchmark datasets of natural document images with all their complexity.

5.6.3 Compared systems

We note that LayTr was trained on a dataset with statistics comparable to the data used for the evaluation of all systems. However, the fonts used in our dataset are standard fonts that are likely to be represented in both the Tesseract and ABBYY training sets. Nevertheless, specific adaptation to the dataset might improve the character-level performance of Tesseract and ABBYY. However, to our knowledge, neither Tesseract nor ABBYY has trainable structure parsing, so in
terms of structure parsing performance, our measured performance arguably represents the best performance that these systems are capable of.

Individual systems for entity detection in the literature achieved higher entity detection than the ABBYY and Tesseract. In this study, we limited our experiments to full end-to-end OCR systems that are capable of processing the data featured in LayTr-Columns and LayTr-Tables without need for re-engineering. This requires the ability to parse both multicolumn and table structure inputs, as well as the ability to output both plain, as well as marked-up texts. Of available commercial systems, we found ABBYY to be of representative performance, while typically outperforming open-source alternatives.

5.6.4 Future work

In future work, LayTr could be extended and trained on full-page images. This would enable evaluation on full-page benchmark datasets and could allow its application on a larger range of real-world inputs.

Unlike previous document OCR systems, LayTr requires only input images and output text as training data. This greatly simplifies the labeling process, since no additional annotations are required for individual entities or document structures. This facilitates large-scale pre-training on new datasets, which should be the subject of further work.

5.7 Summary

This chapter introduced LayTr, a trainable system for joint end-to-end OCR on documents. Our experiments show that LayTr outperforms state-of-the-art systems for end-to-end document OCR. Through the contribution of task-specific datasets for training and evaluation, we show the strong performance of LayTr in both structure parsing and OCR stages.

Existing methods largely treat structure parsing and text recognition stages separately. Through the unified model design and the simple training protocol, LayTr can be efficiently adapted to new data. Furthermore, by combining the two stages, LayTr has the ability to dynamically incorporate geometric and linguistic information, depending on the nature of the document.
CONCLUSIONS AND FUTURE WORK

In this thesis, we contributed towards answering the following three research questions:

1. How can we build holistic systems for full hierarchical document parsing systems?
2. How can we enable more effective and flexible systems for complex real-world settings?
3. How can we overcome the challenges imposed by the separation of structure parsing and text recognition in OCR?

We addressed Question 1 in Chapter 3. We introduced the system ”DocParser” for full hierarchical document parsing. DocParser employs weak supervision to make use of structured document source files. We showed that our weak supervision can significantly reduce the labeling complexity and improve the performance of fine-tuned document parsing systems. Furthermore, we contributed two datasets that are based on scientific publications, arXivdocs-weak and arXivdocs-target. arXivdocs-weak can be used for large-scale weakly supervised pre-training. arXivdocs-target has been manually annotated and can be used for fine-tuning and evaluation of document parsing systems.

We addressed Question 2 in Chapter 4. We contributed ”DSG”, a system for document parsing that is fully end-to-end trainable. By allowing training of the system on joint entity detection and relation classification, no domain-specific heuristics are required and the system components can benefit from the shared use of available labeled information. We furthermore contributed the novel E-Periodica dataset that contains fully annotated document structures for complex magazine pages. We showed that DSG achieves state-of-the-art performance on the document parsing datasets and can be readily applied to create structured hOCR document files for downstream processing tasks.
Chapter 6. Conclusions and Future Work

We addressed Question 3 in Chapter 5. Here, we employ ”LayTr”, a system for joint structure parsing and text recognition. The underlying transformer model can be trained on the OCR task in a unified manner and can make use of both visual information and underlying language model. Since both tasks are commonly performed and evaluated separately, we create two synthetic datasets and evaluation protocols for LayTr, which handles structure parsing implicitly. We assess the performance of both layout analysis and text recognition and show that LayTr outperforms two SOTA OCR engines for both tasks in our experiments.

In future work, unified end-to-end systems for structure parsing and document OCR present a promising research direction. For this, the weak supervision mechanisms outlined in Chapter 3 could be extended to even larger collections of structured document source files to facilitate large-scale model pre-training. Our experiments in Chapter 4 showed that joint end-to-end training allows for more effective application of document parsing systems in real-world settings. With LayTr, we provided a proof-of-concept system that takes this one step further and combines document parsing and text recognition into a unified system. In future work, we believe that the adaptation of LayTr to full-scale documents, combined with large-scale pre-training, could allow for highly effective systems that are easy to train and apply.

With this thesis, we made significant contributions to the development of end-to-end systems for hierarchical structure parsing and document OCR. We first established a holistic system for parsing of hierarchical structures in documents with DocParser. We then further advanced the state-of-the-art in this domain, by not only allowing end-to-end inference, but also full end-to-end training with DSG. Finally, with LayTr, went one step further and jointly integrated hierarchical structure parsing and text recognition in the first fully trainable system for end-to-end document OCR.
A.1 Performance of Document Structure Parsing

A.1.1 Qualitative Evaluation

Figure A.1 shows examples of parsed page structures that are generated by DocParser WS+FT. We illustrate the effects of our structure-based refinement in Figure A.2 and Figure A.3. We observe that bounding boxes of parent entities from the raw predictions are refined such that they fully enclose all of their classified child entities. We particularly achieve improvement of the resulting predicted structure. For instance, for multi-figures, our refinement encloses figure graphics into individual figure structures to match the defined document grammar (see Figure A.2). Figure A.3 shows how two nested entities of the heading category are merged into a single entity during refinement.

We furthermore investigate how the F1 measure for relation classification relates to overall parsing quality. Figure A.4 depicts the detected entities and relations for a document with an F1 score of 0.267. We note that the overall quality of the parsed page is still high. Our relation classification requires entities in the page graph to be exactly matched with the corresponding entities in the ground truth by surpassing the IoU threshold. For instance, the detected header entities are not matched with the ground truth, due to the shape mismatch. This causes a penalty to the F1 score, as several relation triples in the prediction that involve the headings are considered mismatches. Figure A.5 shows another prediction with a low F1 score of 0.417. Here, mismatches can be accounted to the interpretation of entities that could be considered ambigu-
Figure A.1: Qualitative results of DocParser for two samples (top and bottom rows).

ous. For instance, DocParser detects an inline heading in the last content block, while this text segment is interpreted as standard text in the ground truth. We additionally compare our results qualitatively to a state-of-the-art OCR software.\footnote{We compare to outputs of ABBYY Finereader 15.} We observe that the page region detection fails to differentiate between many of the considered semantic categories, e.g. \texttt{heading} \texttt{header} and \texttt{keywords} in Figure A.4 or \texttt{equation} in Figure A.5. We note that the OCR software has access to the original PDF files of full resolution and all meta information, while DocParser only operates on document renderings.
A.1. Performance of Document Structure Parsing

![Diagram](image)

(a) Ground truth  (b) Raw pred.  (c) Predictions

**Figure A.2:** Raw predictions and structure-based refinement in DocParser.

### A.1.2 Reproducibility

For reproducibility purposes, we report results of DocParser on the validation set. Table A.1 and Table A.3 show the performance of the variants of DocParser for entity detection and prediction of hierarchical relations, respectively. Additionally, we include the complete results (including for IoU=0.8) on the test set in Table A.2 and Table A.4.

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.
A.2 Document Grammar

Hierarchical relations between entity pairs follow a predefined grammar (see Table A.5). All entities with meta-information have no ordering, i.e., their relation type is \( \Psi = \text{null} \). 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 \( (E', E, \text{parent_of}) \) with an entity \( E \neq E' \).
Figure A.4: Output with low F1 score (0.267), compared to state-of-the-art OCR software. Green regions in the OCR page recognition correspond to “text areas”. Using the OCR tool, we convert the page to HTML and use our tree-graph to represent the resulting structure. The affiliation section is converted into a list by the OCR software during conversion to HTML.

A.3 Datasets with Document Structure: arXivdocs-target

Figure A.6 shows the number of leaf nodes in the document graph. Furthermore, Table A.6 reports the frequency and average depth of the different entities in the dataset.

Annotators are given a set of instructions for annotating entities: All bounding boxes should fully enclose the contained contents and at most extend to full column width. Figure graphics or captions should always be enclosed by a FIGURE entity. If a figure contains multiple subfigures,
Figure A.5: Output with low F1 score (0.417), compared to state-of-the-art OCR software. Green and red regions in the OCR page recognition correspond to “text areas” and “picture areas”, respectively. Using the OCR tool, we convert the page to HTML and use our tree-graph to represent the resulting structure. The content block in the second “div” section corresponds to the full figure caption text.

Each subfigure should consist of an individual nested figure that contains a figure graphic. Furthermore content block or bibliography block entities are text or bibliography regions that should extend at most a single column/page and contains no other categories. To give annotators the freedom to handle the large variety of document appearances, we do not enforce a strict document grammar during manual annotation.

<table>
<thead>
<tr>
<th>Component 4: Structure-Based Refinement</th>
</tr>
</thead>
<tbody>
<tr>
<td>IoU=0.5</td>
</tr>
<tr>
<td>AP</td>
</tr>
<tr>
<td>mAP</td>
</tr>
<tr>
<td>abstract</td>
</tr>
<tr>
<td>affiliation</td>
</tr>
<tr>
<td>author</td>
</tr>
<tr>
<td>bib. block</td>
</tr>
<tr>
<td>cont. block</td>
</tr>
<tr>
<td>date</td>
</tr>
<tr>
<td>equation</td>
</tr>
<tr>
<td>fig. caption</td>
</tr>
<tr>
<td>fig. graphic</td>
</tr>
<tr>
<td>figure</td>
</tr>
<tr>
<td>footer</td>
</tr>
<tr>
<td>header</td>
</tr>
<tr>
<td>heading</td>
</tr>
<tr>
<td>item</td>
</tr>
<tr>
<td>itemize</td>
</tr>
<tr>
<td>keywords</td>
</tr>
<tr>
<td>page nr.</td>
</tr>
<tr>
<td>tab. caption</td>
</tr>
<tr>
<td>table</td>
</tr>
<tr>
<td>tabular</td>
</tr>
</tbody>
</table>

**Table A.1:** Validation set: Comparison of entity detection (average precision) without structure-based refinement.

**A.4 Datasets with Document Structure: arXivdocs-weak**

Figure A.7 and Table A.7 show the descriptive statistics of the dataset. 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 entities, such as abstract, captions, or content blocks.

**Component 4: Structure-Based Refinement**


Appendix A. DocParser

Table A.2: Test set: Average precision (AP) of entity detection.

<table>
<thead>
<tr>
<th></th>
<th>IoU=0.5</th>
<th></th>
<th>IoU=0.65</th>
<th></th>
<th>IoU=0.8</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Baseline</td>
<td>WS</td>
<td>WS+FT</td>
<td>Baseline</td>
<td>WS</td>
<td>WS+FT</td>
</tr>
<tr>
<td>mean AP</td>
<td>49.9</td>
<td>34.6</td>
<td><strong>69.4</strong></td>
<td>38.5</td>
<td>32.4</td>
<td><strong>56.5</strong></td>
</tr>
<tr>
<td>abstract</td>
<td><strong>95.2</strong></td>
<td>90.5</td>
<td>95.2</td>
<td>90.5</td>
<td>81.0</td>
<td><strong>95.2</strong></td>
</tr>
<tr>
<td>affiliation</td>
<td><strong>51.6</strong></td>
<td>0.0</td>
<td>46.0</td>
<td>5.9</td>
<td>0.0</td>
<td><strong>16.2</strong></td>
</tr>
<tr>
<td>author</td>
<td>18.0</td>
<td>0.0</td>
<td><strong>23.6</strong></td>
<td><strong>20.4</strong></td>
<td>0.0</td>
<td>16.7</td>
</tr>
<tr>
<td>bib. block</td>
<td>42.4</td>
<td>79.1</td>
<td><strong>94.7</strong></td>
<td>43.2</td>
<td><strong>93.9</strong></td>
<td>80.3</td>
</tr>
<tr>
<td>cont. block</td>
<td><strong>89.3</strong></td>
<td>69.8</td>
<td>88.4</td>
<td>83.2</td>
<td>67.0</td>
<td><strong>84.4</strong></td>
</tr>
<tr>
<td>date</td>
<td>0.0</td>
<td>0.0</td>
<td><strong>24.1</strong></td>
<td>0.0</td>
<td>0.0</td>
<td><strong>9.3</strong></td>
</tr>
<tr>
<td>equation</td>
<td>65.8</td>
<td>54.5</td>
<td><strong>82.1</strong></td>
<td>40.6</td>
<td>52.1</td>
<td><strong>72.8</strong></td>
</tr>
<tr>
<td>fig. caption</td>
<td>47.8</td>
<td>30.5</td>
<td><strong>69.2</strong></td>
<td>44.0</td>
<td>17.7</td>
<td><strong>59.5</strong></td>
</tr>
<tr>
<td>fig. graphic</td>
<td>22.3</td>
<td>5.2</td>
<td><strong>60.2</strong></td>
<td>15.9</td>
<td>4.4</td>
<td><strong>54.5</strong></td>
</tr>
<tr>
<td>figure</td>
<td>47.8</td>
<td>35.3</td>
<td><strong>63.5</strong></td>
<td>44.0</td>
<td>33.9</td>
<td><strong>59.4</strong></td>
</tr>
<tr>
<td>footer</td>
<td>55.7</td>
<td>0.0</td>
<td><strong>69.3</strong></td>
<td>48.9</td>
<td>0.0</td>
<td><strong>59.7</strong></td>
</tr>
<tr>
<td>header</td>
<td>79.7</td>
<td>0.0</td>
<td><strong>88.3</strong></td>
<td>64.8</td>
<td>0.0</td>
<td>56.6</td>
</tr>
<tr>
<td>heading</td>
<td>53.7</td>
<td>52.1</td>
<td><strong>66.4</strong></td>
<td>33.1</td>
<td><strong>46.0</strong></td>
<td>45.4</td>
</tr>
<tr>
<td>item</td>
<td>0.0</td>
<td>33.6</td>
<td><strong>50.5</strong></td>
<td>0.0</td>
<td><strong>35.3</strong></td>
<td>33.5</td>
</tr>
<tr>
<td>itemize</td>
<td>0.0</td>
<td>25.0</td>
<td><strong>58.3</strong></td>
<td>0.0</td>
<td>25.0</td>
<td><strong>50.0</strong></td>
</tr>
<tr>
<td>keywords</td>
<td>36.4</td>
<td>0.0</td>
<td><strong>59.0</strong></td>
<td>36.4</td>
<td>0.0</td>
<td><strong>43.0</strong></td>
</tr>
<tr>
<td>page nr.</td>
<td>74.7</td>
<td>0.0</td>
<td><strong>77.3</strong></td>
<td>28.5</td>
<td>0.0</td>
<td><strong>42.0</strong></td>
</tr>
<tr>
<td>tab. caption</td>
<td>55.2</td>
<td>69.1</td>
<td><strong>76.6</strong></td>
<td>40.2</td>
<td>61.6</td>
<td><strong>63.4</strong></td>
</tr>
<tr>
<td>table</td>
<td>84.5</td>
<td><strong>96.3</strong></td>
<td>94.3</td>
<td>62.7</td>
<td>87.9</td>
<td><strong>89.6</strong></td>
</tr>
<tr>
<td>tabular</td>
<td>78.4</td>
<td>50.8</td>
<td><strong>100.0</strong></td>
<td>68.4</td>
<td>42.4</td>
<td><strong>99.5</strong></td>
</tr>
</tbody>
</table>

(4) In our experiments, we use \( r = 30 \) for structure-based refinement. During development, we observed only minor differences for values of \( r = 10 \) and higher. To confirm this, we analyze the performance of DocParser WS+FT for \( r = 2, 5, 10, 20, 30 \) on the validation set (see Table A.8).\(^2\) Here we observe that the accuracy of our system remains unchanged for values of \( r \geq 10 \).

\(^2\)Note that results are given for a single model and can differ from the detailed relation classification evaluation, where we average over three models.
### Table A.3: Validation set: Performance in predicting hierarchical relations (as measured by F1).

<table>
<thead>
<tr>
<th></th>
<th>IoU=0.5</th>
<th></th>
<th>IoU=0.65</th>
<th></th>
<th>IoU=0.8</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Baseline</td>
<td>WS</td>
<td>WS+FT</td>
<td>Baseline</td>
<td>WS</td>
<td>WS+FT</td>
</tr>
<tr>
<td>All</td>
<td>0.406</td>
<td>0.369</td>
<td><strong>0.550</strong></td>
<td>0.337</td>
<td>0.350</td>
<td><strong>0.478</strong></td>
</tr>
<tr>
<td>followed_by</td>
<td>0.360</td>
<td>0.368</td>
<td><strong>0.531</strong></td>
<td>0.303</td>
<td>0.356</td>
<td><strong>0.454</strong></td>
</tr>
<tr>
<td>parent_of</td>
<td>0.519</td>
<td>0.372</td>
<td><strong>0.592</strong></td>
<td>0.418</td>
<td>0.333</td>
<td><strong>0.532</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>IoU=0.5</th>
<th></th>
<th>IoU=0.65</th>
<th></th>
<th>IoU=0.8</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Baseline</td>
<td>WS</td>
<td>WS+FT</td>
<td>Baseline</td>
<td>WS</td>
<td>WS+FT</td>
</tr>
<tr>
<td>All</td>
<td>0.447</td>
<td>0.401</td>
<td><strong>0.658</strong></td>
<td>0.359</td>
<td>0.385</td>
<td><strong>0.576</strong></td>
</tr>
<tr>
<td>followed_by</td>
<td>0.402</td>
<td>0.390</td>
<td><strong>0.602</strong></td>
<td>0.310</td>
<td>0.384</td>
<td><strong>0.513</strong></td>
</tr>
<tr>
<td>parent_of</td>
<td>0.551</td>
<td>0.428</td>
<td><strong>0.771</strong></td>
<td>0.470</td>
<td>0.387</td>
<td><strong>0.704</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>IoU=0.5</th>
<th></th>
<th>IoU=0.65</th>
<th></th>
<th>IoU=0.8</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Baseline</td>
<td>WS</td>
<td>WS+FT</td>
<td>Baseline</td>
<td>WS</td>
<td>WS+FT</td>
</tr>
<tr>
<td>All</td>
<td>0.416</td>
<td>0.343</td>
<td><strong>0.504</strong></td>
<td>0.322</td>
<td>0.318</td>
<td><strong>0.445</strong></td>
</tr>
<tr>
<td>followed_by</td>
<td>0.413</td>
<td>0.387</td>
<td><strong>0.506</strong></td>
<td>0.314</td>
<td>0.366</td>
<td><strong>0.447</strong></td>
</tr>
<tr>
<td>parent_of</td>
<td>0.421</td>
<td>0.235</td>
<td><strong>0.500</strong></td>
<td>0.339</td>
<td>0.198</td>
<td><strong>0.443</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>IoU=0.5</th>
<th></th>
<th>IoU=0.65</th>
<th></th>
<th>IoU=0.8</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Baseline</td>
<td>WS</td>
<td>WS+FT</td>
<td>Baseline</td>
<td>WS</td>
<td>WS+FT</td>
</tr>
<tr>
<td>All</td>
<td>0.453</td>
<td>0.382</td>
<td><strong>0.615</strong></td>
<td>0.363</td>
<td>0.354</td>
<td><strong>0.558</strong></td>
</tr>
<tr>
<td>followed_by</td>
<td>0.455</td>
<td>0.410</td>
<td><strong>0.581</strong></td>
<td>0.351</td>
<td>0.394</td>
<td><strong>0.524</strong></td>
</tr>
<tr>
<td>parent_of</td>
<td>0.451</td>
<td>0.317</td>
<td><strong>0.679</strong></td>
<td>0.389</td>
<td>0.263</td>
<td><strong>0.620</strong></td>
</tr>
</tbody>
</table>

We additionally provide pseudo-code for our refinement procedure in Algorithm 2.

**Component 5: Scalable Weak Supervision**

To analyze the degree of noise in arXivdocs-weak, we evaluate the average precision for the weak annotations against the manually generated ground truth in arXivdocs-target. Table A.9 shows the accuracies of arXivdocs-weak for different IoU values, as measured on the training split of arXivdocs-target. We observe various AP values of 0, indicating the absence of the respective categories in arXivdocs-weak. Furthermore, for the majority of categories, the measured is relatively low (AP \( \leq 0.5 \) for IoU \( \geq 0.5 \)). This emphasizes the systematic noise in
### Table A.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. Hierarchical nesting for the child entities of floats, e.g. captions, is, however, encouraged in the automatic refinement process. Further details are included in the supplements.

arXivdocs-weak and confirms the positioning of our experimental setting in the domain of weak supervision.
A.5. Related Work

A.5.1 Weak Supervision for Document Layout:

(Zhong et al., 2019) (PN) use weak supervision for detection of page layout entities. The dataset features 5 coarse categories, compared to 23 fine-grained categories in arXivdocs. Furthermore, the system does not contain a relation classification component. Following, we examine differ-
Algorithm 2  Structure-based refinement

Input: Detected Entities

Output: Refined Entities; Hierarchical Relations

1: function REFINEENTITIES(Detected entities)
   counter = 0;
2:   while counter ≤ θ do
3:     if (1) Parent entity bounding boxes don’t fully enclose children then
4:         Expand parent bounding boxes s.t. they enclose children;
          Go to start of loop;
5:     if (2) Directly nested entities of same category exist then
6:         Merge directly nested entities into a single entity;
          Go to start of loop;
7:     if (3) Siblings found that are not allowed to co-exist in hierarchy then
8:         Enclose groups of siblings with new, valid parent entities;
          Go to start of loop;
9:     if (4) Possible parent found in neighborhood of a parent-less entity then
10:    Expand matched parents bounding boxes to enclose child entities;
         Go to start of loop;
        Exit loop;
   Classify all hierarchical relations;

ences and correspondences between the five classes in (Zhong et al., 2019) and our arXivdocs:

- TEXT: Corresponds to content block in arXivdocs. In contrast to our dataset, the TEXT category corresponds to individual paragraphs (instead of uninterrupted text on a single column) and is used for captions.

- TITLE: corresponds to our HEADER category.

- LIST: corresponds to our ITEMIZE category. A difference here is that LIST entities in PN are separated by columns.

- TABLE: corresponds to our TABULAR. In contrast to arXivdocs, they do not feature nesting relations that contain, for instance, TABLE CAPTION entities. Fine-grained children, such as cells, rows and columns are also not featured.

- FIGURE: corresponds roughly to the concept of FIGURE GRAPHIC in arXivdocs. However, no nesting relations (i.e. sub-figures) or captions are featured.
We evaluate the feasibility of using the dataset presented in PN for pre-training. We use the same pre-training procedure as in our experiments that utilize arXivdocs. To account for the difference of pre-training and target domains, we use an extended fine-tuning procedure of PN that matches the pre-training scheme of up to 80,000 iterations. Table A.10 shows results for entity detection. Here we observe that pre-training improves the performance of the system, when compared to DocParser Baseline that does not use weak supervision. We also observe that pre-training with the PN dataset results in significantly lower mAP values, e.g. 60.0 after fine-tuning compared to 69.4 in DocParser WS+FT at IoU = 0.5. For some entity categories, we observe higher individual AP values for PN, e.g. affiliation at IoU = 0.5. We attribute this to the higher occurrence of more compact text entities in PN. Additionally, this could also be caused by our experimental protocol in which early stopping is applied function of the mAP value, instead of individual AP values. As such, there is a performance trade-off between individual entity categories.

A.6 Robustness Check: Table Structure Parsing

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

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.

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 \( \gamma = \frac{\text{area}(B_{\text{cell}} \cap B_{\text{text}})}{\text{area}(B_{\text{text}})} \), using \( \gamma \geq 0.5 \).

A.6.1 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. Row 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.
A.6. Robustness Check: Table Structure Parsing

A.6.2 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, we follow (Schreiber et al., 2017) 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. Afterwards, we move the borders of all pairs of neighboring rows to their respective midpoint. Analogously, we adjust all column bounding
Figure A.8: Comparison of test mAP (IoU=0.5) for three variants of DocParser for detection of table structure annotations. We follow the same procedure as described for fine-tuning with the default document entities. The weakly supervised system DocParser WS outperforms the baseline system without fine-tuning (FT). Fine-tuning with 10 or more images yields additional performance gains.

boxes. Cell bounding boxes are newly created from row and column intersections in a final step.

A.6.2.1 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 gg table rows and columns and the impact of fine-tuning are shown in Figure A.8. Compared to full document pages, we measure higher mAPs for all systems. We observe that the weakly supervised model outperforms DocParser Baseline 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.
### A.6. Robustness Check: Table Structure Parsing

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

**Table A.6:** Statistics by entity of arXivdocs-target.
### Appendix A. DocParser

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

**Table A.7:** Summary statistics by entity of arXivdocs-weak dataset.
### A.6. Robustness Check: Table Structure Parsing

Table A.8: Impact of $r$ on the relation classification performance on the development set for a DocParser WS+FT model.

<table>
<thead>
<tr>
<th>$r$</th>
<th>IoU=0.5</th>
<th>IoU=0.65</th>
<th>IoU=0.8</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.619</td>
<td>0.563</td>
<td>0.384</td>
</tr>
<tr>
<td>5</td>
<td>0.680</td>
<td>0.619</td>
<td>0.427</td>
</tr>
<tr>
<td>10</td>
<td>0.680</td>
<td>0.619</td>
<td>0.427</td>
</tr>
<tr>
<td>20</td>
<td>0.680</td>
<td>0.619</td>
<td>0.427</td>
</tr>
<tr>
<td>30</td>
<td>0.680</td>
<td>0.619</td>
<td>0.427</td>
</tr>
</tbody>
</table>
Table A.9: Average precision (AP) of entities in arXivdocs-weak, compared to the training split of arXivdocs-target

<table>
<thead>
<tr>
<th>Entity</th>
<th>IoU=0.5</th>
<th>IoU=0.65</th>
<th>IoU=0.8</th>
</tr>
</thead>
<tbody>
<tr>
<td>mAP</td>
<td>30.0</td>
<td>25.5</td>
<td>19.4</td>
</tr>
<tr>
<td>abstract</td>
<td>61.1</td>
<td>43.5</td>
<td>9.8</td>
</tr>
<tr>
<td>affiliation</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>author</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>bib. block</td>
<td>47.6</td>
<td>47.6</td>
<td>31.6</td>
</tr>
<tr>
<td>cont. block</td>
<td>38.9</td>
<td>32.7</td>
<td>23.2</td>
</tr>
<tr>
<td>date</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>equation</td>
<td>26.2</td>
<td>24.5</td>
<td>22.8</td>
</tr>
<tr>
<td>fig. caption</td>
<td>24.9</td>
<td>23.8</td>
<td>23.8</td>
</tr>
<tr>
<td>fig. graphic</td>
<td>18.5</td>
<td>14.7</td>
<td>14.7</td>
</tr>
<tr>
<td>figure</td>
<td>26.5</td>
<td>19.5</td>
<td>11.2</td>
</tr>
<tr>
<td>footer</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>header</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>heading</td>
<td>33.6</td>
<td>33.6</td>
<td>33.6</td>
</tr>
<tr>
<td>item</td>
<td>25.1</td>
<td>9.6</td>
<td>4.8</td>
</tr>
<tr>
<td>itemize</td>
<td>54.5</td>
<td>42.4</td>
<td>42.4</td>
</tr>
<tr>
<td>keywords</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>page nr.</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>tab. caption</td>
<td>76.3</td>
<td>76.3</td>
<td>72.7</td>
</tr>
<tr>
<td>table</td>
<td>74.7</td>
<td>60.5</td>
<td>44.6</td>
</tr>
<tr>
<td>tabular</td>
<td>91.7</td>
<td>80.8</td>
<td>52.1</td>
</tr>
</tbody>
</table>
### A.6. Robustness Check: Table Structure Parsing

<table>
<thead>
<tr>
<th>AP</th>
<th>IoU=0.5</th>
<th>IoU=0.65</th>
<th>IoU=0.8</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean AP</td>
<td>8.4</td>
<td>60.0</td>
<td>3.8</td>
</tr>
<tr>
<td>abstract</td>
<td>0.0</td>
<td>89.2</td>
<td>0.0</td>
</tr>
<tr>
<td>affiliation</td>
<td>0.0</td>
<td>63.8</td>
<td>0.0</td>
</tr>
<tr>
<td>author</td>
<td>0.0</td>
<td>34.3</td>
<td>0.0</td>
</tr>
<tr>
<td>bib. block</td>
<td>0.0</td>
<td>60.3</td>
<td>0.0</td>
</tr>
<tr>
<td>cont. block</td>
<td>34.9</td>
<td>90.3</td>
<td>20.1</td>
</tr>
<tr>
<td>date</td>
<td>0.0</td>
<td>16.7</td>
<td>0.0</td>
</tr>
<tr>
<td>equation</td>
<td>0.0</td>
<td>77.3</td>
<td>0.0</td>
</tr>
<tr>
<td>fig. caption</td>
<td>0.0</td>
<td>64.8</td>
<td>0.0</td>
</tr>
<tr>
<td>fig. graphic</td>
<td>0.0</td>
<td>38.4</td>
<td>0.0</td>
</tr>
<tr>
<td>figure</td>
<td>23.0</td>
<td>49.9</td>
<td>6.0</td>
</tr>
<tr>
<td>footer</td>
<td>0.0</td>
<td>69.1</td>
<td>0.0</td>
</tr>
<tr>
<td>header</td>
<td>0.0</td>
<td>76.9</td>
<td>0.0</td>
</tr>
<tr>
<td>heading</td>
<td>27.4</td>
<td>63.8</td>
<td>12.3</td>
</tr>
<tr>
<td>item</td>
<td>0.0</td>
<td>1.7</td>
<td>0.0</td>
</tr>
<tr>
<td>itemize</td>
<td><strong>36.5</strong></td>
<td>25.0</td>
<td><strong>12.5</strong></td>
</tr>
<tr>
<td>keywords</td>
<td>0.0</td>
<td>50.0</td>
<td>0.0</td>
</tr>
<tr>
<td>page nr.</td>
<td>0.0</td>
<td>80.3</td>
<td>0.0</td>
</tr>
<tr>
<td>tab. caption</td>
<td>0.0</td>
<td>62.7</td>
<td>0.0</td>
</tr>
<tr>
<td>table</td>
<td>46.5</td>
<td>90.8</td>
<td>25.0</td>
</tr>
<tr>
<td>tabular</td>
<td>0.0</td>
<td>94.8</td>
<td>0.0</td>
</tr>
</tbody>
</table>

**Table A.10:** Average precision (AP) of entity detection on the test set, using (Zhong et al., 2019) (PN) and structure-based refinement.
B.1 DSG System

B.1.1 Component 4: Grammar-based postprocessing

We give further details for the rules applied in our postprocessing as follows.

**g\_Il** Illegal relations:

- We enforce that the root entity of type doc. root can only be part of relations in which it is a parent entity.

- All relations are anti-symmetric. For this, we ensure that no two relations with \( \Psi \in \{\text{parent\_of, followed\_by}\} \) exist that result in a symmetric relation. We resolve such cases by deleting the conflicting relations with lower confidence.

- No two relations with the same \( \Psi \in \{\text{followed\_by, parent\_of}\} \) end in the same entity.

- We ensure that entities can only be followed by at most one other entity.

- We ensure that entity of the \texttt{UNORDERED GROUP} category are not part of any sequential relations.

- We remove any cycles that might be formed by the graph formed by the predicted relations.
Sequential relations can only exist between sibling entities belonging to the same parent entity.

$g_{\text{mis}}$ **Missing relations:** If, after performing the previous postprocessing steps an entity does not have a parent entity, we inspect the confidence scores of all relations $(E_{\text{cand}} \cdot E_{\text{missing-parent}} \cdot \Psi)$ with $\Psi = \text{parent}_{of}$ and candidate parent entities $E_{\text{cand}}$. We retrieve the relation with the highest confidence score, even if this score would otherwise not be sufficiently high to determine a relation with $\Psi = \text{parent}_{of}$ and ensure that the resulting relation adheres to $g_{\text{lg}}$.

## B.2 Datasets

### B.2.1 E-Periodica

![Figure B.1](image)

**Figure B.1:** Distribution of the number of issues per magazine in the E-Periodica dataset.

Figure B.2 shows the distribution of page types in the E-Periodica dataset. The distribution of the number of issues per magazine is shown in Figure B.1.

A ground-truth annotation set was assembled for E-Periodica in the first annotation phase, using an extended set of 64 available entity categories (see Table B.1. To allow for more consistent
downstream processing and balanced categories, this first version of annotations is aggregated down to 23 entity categories. Aggregation is performed by mapping classes $\varphi$ with fewer than 45 occurrences to their closest match $\mu$ for training purposes. A list of $(\varphi, \mu)$ tuples denoting the mapping is provided in Table B.2.

### B.3 Evaluation

#### B.3.1 Qualitative evaluation

Figures B.4 and B.5 show additional qualitative examples. For easier assessment, we feature unedited input images again in Figures B.3a and B.3b.

![Figure B.2: Distribution of page types in the E-Periodica dataset.](image)
Appendix B. Document Structure Generator

(a) Example of document with 4 UNORDERED GROUP entities.

(b) Example of document with multiple articles.

Figure B.3: Examples of documents featured in E-Periodica.
### Table B.1: Statistics of original 64 categories available in the tagging of E-Periodica (frequency zero refers to this category not being chosen by our annotators in the process).
### Table B.2: Mapping of E-Periodica entity categories from first annotation phase to final categories.

<table>
<thead>
<tr>
<th>E-Periodica v1</th>
<th>E-Periodica final</th>
<th>E-Periodica v1</th>
<th>E-Periodica final</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOC. ROOT</td>
<td>DOC. ROOT</td>
<td>FRONT MATTER</td>
<td>ARTICLE</td>
</tr>
<tr>
<td>META</td>
<td>META</td>
<td>GROUP</td>
<td>ARTICLE</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>TEXT BLOCK</td>
<td>HEADER</td>
<td>HEADER</td>
</tr>
<tr>
<td>ADVERTISING</td>
<td>ARTICLE</td>
<td>HEADING</td>
<td>HEADING</td>
</tr>
<tr>
<td>AFFILIATION</td>
<td>TEXT BLOCK</td>
<td>INDEX</td>
<td>ARTICLE</td>
</tr>
<tr>
<td>APPENDIX</td>
<td>ARTICLE</td>
<td>INTRODUCTION</td>
<td>TEXT BLOCK</td>
</tr>
<tr>
<td>ARTICLE</td>
<td>ARTICLE</td>
<td>ITEM</td>
<td>ITEM</td>
</tr>
<tr>
<td>NEWS</td>
<td>ARTICLE</td>
<td>ITEM COLUMN</td>
<td>ITEM</td>
</tr>
<tr>
<td>AUTHOR</td>
<td>AUTHOR</td>
<td>ITEMIZE</td>
<td>ITEMIZE</td>
</tr>
<tr>
<td>BACKGR. FIG.</td>
<td>BACKGR. FIG.</td>
<td>LINE</td>
<td>TEXT BLOCK</td>
</tr>
<tr>
<td>BACK MATTER</td>
<td>ARTICLE</td>
<td>LOGO</td>
<td>FIG. GRAPHIC</td>
</tr>
<tr>
<td>BIBL. BLOCK</td>
<td>ITEMIZE</td>
<td>MISCELLANEOUS</td>
<td>ARTICLE</td>
</tr>
<tr>
<td>BIBLIOGRAPHY</td>
<td>ITEMIZE</td>
<td>ORDERED GROUP</td>
<td>ORDERED GROUP</td>
</tr>
<tr>
<td>BOOK REVIEW</td>
<td>ARTICLE</td>
<td>PAGE NR.</td>
<td>PAGE NR.</td>
</tr>
<tr>
<td>CHAPTER</td>
<td>ARTICLE</td>
<td>PARAGRAPH</td>
<td>TEXT BLOCK</td>
</tr>
<tr>
<td>CODE</td>
<td>TEXT BLOCK</td>
<td>POSTFACE</td>
<td>ARTICLE</td>
</tr>
<tr>
<td>COMPETITIONS</td>
<td>ARTICLE</td>
<td>PREFACE</td>
<td>ARTICLE</td>
</tr>
<tr>
<td>TEXT BLOCK</td>
<td>TEXT BLOCK</td>
<td>QUOTE</td>
<td>TEXT BLOCK</td>
</tr>
<tr>
<td>TEXT LINE</td>
<td>TEXT BLOCK</td>
<td>REFERENCE LIST</td>
<td>ARTICLE</td>
</tr>
<tr>
<td>TEXT LINES</td>
<td>TEXT BLOCK</td>
<td>SIDE</td>
<td>TEXT BLOCK</td>
</tr>
<tr>
<td>DATE</td>
<td>TEXT BLOCK</td>
<td>SUBJECT</td>
<td>TEXT BLOCK</td>
</tr>
<tr>
<td>EQUATION</td>
<td>TEXT BLOCK</td>
<td>TABLE CAPTION</td>
<td>TEXT BLOCK</td>
</tr>
<tr>
<td>EQ. FORMULA</td>
<td>TEXT BLOCK</td>
<td>TABLE</td>
<td>TABLE</td>
</tr>
<tr>
<td>EQ. LABEL</td>
<td>TEXT BLOCK</td>
<td>TABLE CELL</td>
<td>ROW</td>
</tr>
<tr>
<td>FIGURE</td>
<td>FIGURE</td>
<td>TABLE COLUMN</td>
<td>COL</td>
</tr>
<tr>
<td>FIG. GRAPHIC</td>
<td>FIG. GRAPHIC</td>
<td>TABLE FOOTNOTE</td>
<td>FOOTNOTE</td>
</tr>
<tr>
<td>FIG. CAPTION</td>
<td>FIG. CAPTION</td>
<td>TABLE ROW</td>
<td>ROW</td>
</tr>
<tr>
<td>FOOT</td>
<td>FOOT</td>
<td>TABLE OF CONT.</td>
<td>TABLE OF CONT.</td>
</tr>
<tr>
<td>FOOTNOTE</td>
<td>FOOTNOTE</td>
<td>TABULAR</td>
<td>TABULAR</td>
</tr>
<tr>
<td>FORM</td>
<td>ITEMIZE</td>
<td>UNORD. GROUP</td>
<td>UNORD. GROUP</td>
</tr>
</tbody>
</table>
Figure B.4: Qualitative evaluation comparing the parsed hierarchical document structure by different 
systems. Document is characterized by fairly flat but long structure. Top: entity recognition; bottom: 
hierarchical structure.
Figure B.5: Qualitative evaluation comparing the parsed hierarchical document structure by different systems. The document is characterized by multiple, separate advertisements. Top: entity recognition; bottom: hierarchical structure.
BIBLIOGRAPHY


Emilia Apostolova and Noriko Tomuro. 2014. Combining visual and textual features for information extraction from online flyers. In Conference on Empirical Methods in Natural Language Processing (EMNLP).


Bibliography


Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.


Bibliography


Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, Ming Zhou, and Zhoujun Li. 2020b. TableBank: A Benchmark Dataset for Table Detection and Recognition. In *Conference on Language Resources and Evaluation* (LREC).

Minghao Li, Tengchao Lv, Jingye Chen, Lei Cui, Yijuan Lu, Dinei Florencio, Cha Zhang, Zhoujun Li, and Furu Wei. 2022b. TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models.


Xiaojing Liu, Feiyu Gao, Qiong Zhang, and Huasha Zhao. 2019a. Graph convolution for multimodal information extraction from visually rich documents. In Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT).


131


Johannes Rausch, Ce Zhang, and Thomas Breuel. 2023b. LayTr: Differentiable end-to-end OCR for Complex Layouts with Transformers. Under review for ICDAR.


Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. 2020. LayoutLM: Pre-training of Text and Layout for Document Image Understanding. In *International Conference on Knowledge Discovery & Data Mining*.

Burcu Yıldız, Katharina Kaiser, and Silvia Miksch. 2005. Pdf2table: A Method to Extract Table Information from PDF Files. *Indian International Conference on Artificial Intelligence (IICAI)*.


