

DSG: An End-to-End Document Structure Generator

Anonymous Authors

organizations

City, Country

email address

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 a structured hierarchical format. However, existing systems for this task are limited by heuristics and are *not end-to-end trainable*. In this work, we introduce the *Document Structure Generator* (**DSG**), a novel system for document parsing that is fully *end-to-end trainable*. **DSG** combines 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. Unlike existing systems that rely on heuristics, our **DSG** is trained *end-to-end*, making it effective and flexible for real-world applications. We further contribute a new, large-scale dataset called E-Periodica comprising real-world magazines with complex document structures for evaluation. Our results demonstrate that our **DSG** outperforms commercial OCR tools and, on top of that, achieves state-of-the-art performance. To the best of our knowledge, our **DSG** system is the first *end-to-end trainable* system for hierarchical document parsing.

Index Terms—Information Extraction, Parsing; Data Mining, Document Analysis

I. INTRODUCTION

Large 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 [1]. This is a crucial hurdle 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 [2], [3]. 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 [4], [5], [6], [7], [8] and knowledge base construction [9]. To this end, there is a direct need in practice [10] 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 (see Fig. 1). 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 [11] 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. Yet, the task is challenging due to the complex, nested structures in real-world documents.

A standard solution to map document renderings onto parsable documents are optical character recognition (OCR) systems. Current OCR systems are highly effective in retrieving word-level textual contents from rendered documents [12]. However, OCR systems generally focus only on the textual contents but struggle with inferring the hierarchical structure. OCR systems generally rely upon a prior step for parsing the document structure, yet which is still very challenging and error-prone [10]. As a consequence, OCR systems suffer from large errors, especially if errors in the step for parsing the structure parsing occur [13]. To address this, earlier research focused on custom systems for parsing specific entities in documents such as table structures [14], [15] but without parsing the complete hierarchical structure in documents. Even other research aimed at identifying document entities [16], [17], [18], but without actually generating hierarchical document structures. Only one work is tailored to generate hierarchical document structures from document renderings [11]. 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*.

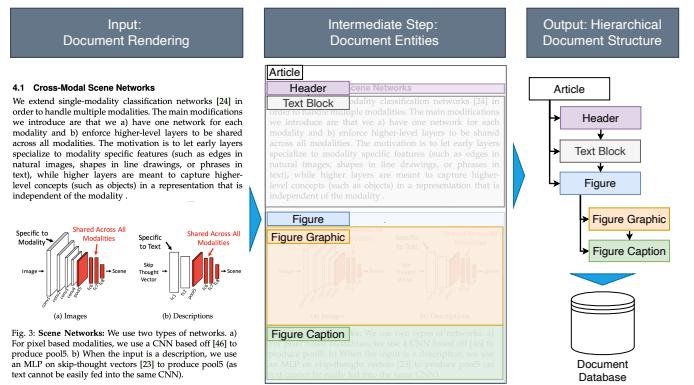


Fig. 1: The task of generating document structures by identifying (i) entities within documents and (ii) relations describing the hierarchical structure.

Our DSG system:¹ We develop the *Document Structure*

¹Codes for our system, online supplements, and dataset are publicly available at <https://github.com/jds2cirwyc2w/DSG>.

Generator (DSG), a novel 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 and evaluating hierarchical document structures called E-Periodica. E-Periodica is based on real-world magazines from different source languages (e.g., English, German, French, Italian). We manually annotated the hierarchical document 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 [11]. 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.

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.

II. RELATED WORK

Document structure parsing: Existing systems have two shortcomings in that they are either (i) limited to entity recognition and thus do *not* generate hierarchical structures, or (ii) 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 [19], [20]. As such, the works extract 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 [21], [22], [23], [14], [24]. In table structure parsing, the aim is to recognize the structure

(e.g., rows, cells) in tables [15], [14]. Here, the input is provided either as text [25], [26] or through document renderings [27]. 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., [17], [18].) 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 [28].

Hierarchical document parsers: One work [29] classifies lines and their immediate parent lines but using of third-party OCR tools in that works leads to error propagation throughout the system. More importantly, the work is limited to a few, high-level entities (e.g., table structures, lists, and sub-figures are missing) and most of the hierarchical structure is lost (e.g., orderings). Hence, this work fails to generate comprehensive document structures as required in our task.

Closest to our work is a system called DocParser [11], 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 [30]. Many recent methods for scene graph generation are based on two-stage training procedures where the detection components build upon Faster R-CNN [31], [32], [33]. 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.

III. PROBLEM DESCRIPTION

Objective: The objective of our system 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, \dots, D_n . The outputs are hierarchically-structured documents given by pairs $(H_1, T_1), \dots, (H_n, T_n)$, where $H_i, i = 1 \dots, n$, captures the hierarchical structure and $T_i, i = 1 \dots, n$, the texts. 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 $E_j, j = 1, \dots, m$, and relations by $R_j, j = 1, \dots, k$. Both are defined below. The above task is thus analogous to earlier research on parsing hierarchical document structures (e.g., [11]).

Entities: Entities capture the different structural elements in documents, such as figures, tables, captions, text blocks, etc. Each entity E_j , $j = 1, \dots, m$, is described by four attributes: (1) a semantic category $c_j \in \mathcal{C} = \{C_1, \dots, C_l\}$ (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 , $j = 1, \dots, k$, are defined by triples $(E_{\text{subj}}, E_{\text{obj}}, \Psi)$ consisting of a subject E_{subj} , an object E_{obj} , and a relation type $\Psi \in \{\text{parent_of}, \text{followed_by}, \text{null}\}$. Furthermore, the relations R_j are associated with a confidence score P_j^Ψ for the predicted relation type Ψ .

hOCR output: As additional output, hierarchically-structured text documents given by pairs $(H_1, T_1), \dots, (H_n, T_n)$, where T_i , $i = 1 \dots, n$, captures the texts should further be provided in the in the standardized hOCR format [34] to facilitate downstream processing tasks. hOCR is a markup language for representing and storing structured documents in a unified format [34]. 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.

IV. OUR DSG SYSTEM

Overview: An overview of our DSG system is shown in Fig. 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.

A. 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 ϕ_s^{\max} . If the longest side would exceed a predefined maximum size of ϕ_l^{\max} after this step, the images are resized so that their longest side length is ϕ_l^{\max} . During training, the image size can be varied for the purpose of data augmentation.² For this, ϕ_s^{\max} and ϕ_l^{\max} are randomly chosen from a set of different sizes. The images are then normalized following the procedure in [35]. Specifically, we subtract the

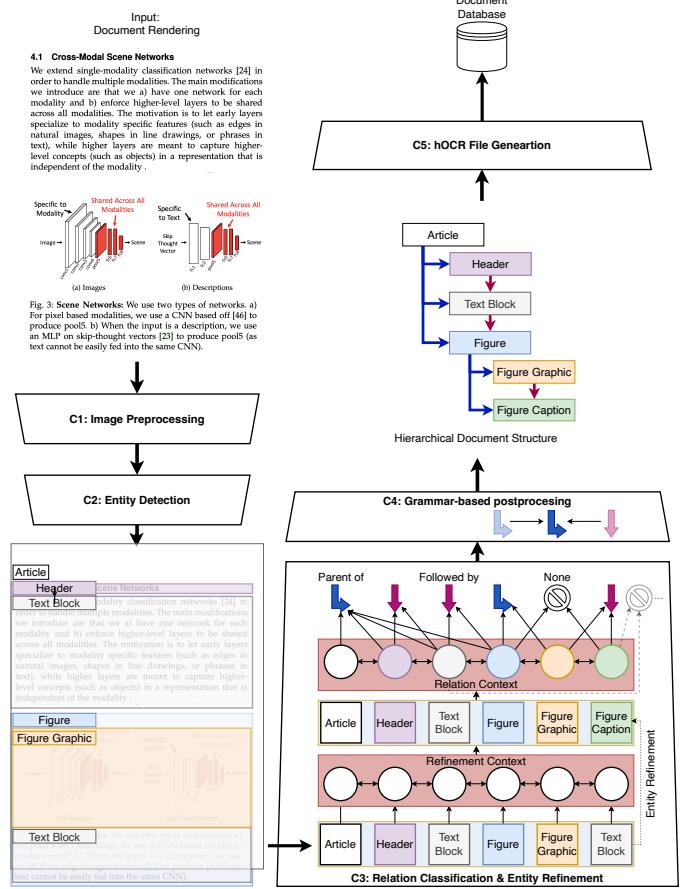


Fig. 2: Overview of our DSG system.

mean channel-wise pixel values of the underlying pre-training dataset [36] from the inputs.

B. Entity detection (C2)

The second component builds upon a Faster R-CNN architecture [37] for entity detection. Here, visual feature maps on different scales are extracted via a convolutional neural network [38], [39]. 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.

C. Relation classification and entity refinement (C3)

The relation classification in DSG builds upon the neural motifs architecture [31]. This architecture extends the entity detection architecture with two additional neural network

²In 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).

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, $(E_{\text{subj}}, E_{\text{obj}})$, to classify if they form a relation triple $(E_{\text{subj}}, E_{\text{obj}}, \Psi)$ of type $\Psi \in \{\text{parent_of}, \text{followed_by}, \text{null}\}$. 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^Ψ for all considered entity pairs $(E_{\text{subj}}, E_{\text{obj}})$. If the respective confidence score exceeds a predefined threshold τ , a relation R_j of type $\Psi \in \{\text{parent_of}, \text{followed_by}, \text{null}\}$ 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 $\rho_i^{\text{ref_in}}$, $i \in \{\text{vis}, \text{cat}, \text{pos}\}$, as follows. First, $\rho_{\text{vis}}^{\text{ref_in}}$ 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, $\rho_{\text{cat}}^{\text{ref_in}}$ 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 [40]. Third, $\rho_{\text{pos}}^{\text{ref_in}}$ 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 to the features $\rho_i^{\text{ref_in}}$, $i \in \{\text{vis}, \text{cat}, \text{pos}\}$ 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* $\rho^{\text{ref_out}}$. 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* $\rho_i^{\text{rel_in}}$, $i \in \{\text{vis}, \text{cat}, \text{ref}\}$ as follows. First, $\rho_{\text{vis}}^{\text{rel_in}}$ is visual feature map, identical to that in $\rho_{\text{vis}}^{\text{ref_in}}$. Second, $\rho_{\text{cat}}^{\text{rel_in}}$ is the category embedding, analogous to the category embedding $\rho_{\text{cat}}^{\text{ref_in}}$ to represent the refined entity category. Third, $\rho_{\text{ref}}^{\text{rel_in}}$ are the refinement context output features, i.e., $\rho_{\text{ref}}^{\text{rel_in}} = \rho^{\text{ref_out}}$.

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* ρ_1^{pair} , ρ_2^{pair} , and ρ_3^{pair} as specified in the following. Later, the pair-wise features are used to predict confidence scores P_j^Ψ for all considered entity pairs $(E_{\text{subj}}, E_{\text{obj}})$. Specifically, the pair-wise features are: First, ρ_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 $B_{\text{pair}} = \text{Union}(E_{\text{subj}}, E_{\text{obj}})$. Second, ρ_2^{pair} is formed for entity-entity pairs $(E_{\text{subj}}, E_{\text{obj}})$ by concatenating the respective refinement context output features $(\rho_{\text{subj}}^{\text{rel_out}}, \rho_{\text{obj}}^{\text{rel_out}})$. Third, a frequency bias term ρ_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 $(E_{\text{subj}}, E_{\text{obj}}, \Psi)$ in the training set. The frequency bias term thus reflects that, for certain pairings of entity categories $(c_{\text{subj}}, c_{\text{obj}})$, relation types $\Psi \in \{\text{parent_of}, \text{followed_by}, \text{null}\}$ are more or less likely. Finally, the pair-wise features ρ_1^{pair} , ρ_2^{pair} , and ρ_3^{pair} are then combined to a pair-wise output feature $\rho_{\text{out}}^{\text{pair}}$ that used to predict the P_j^Ψ 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.

D. Grammar-based postprocessing (C4)

This component of our system converts hierarchical document structures H_i , $i = 1, \dots, n$, 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 [34]. 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^Ψ . In particular, we apply our grammar-based postprocessing in sequential steps to address root entities (g_{rt}), illegal entities (g_{ilg}), and missing relations (g_{mis}) as follows:

- **Root entities** (g_{rt}): We append additional entities to build a basic skeleton for the document files. Specifically, we add root entities **DOC. ROOT**, **ARTICLE** and **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.
- **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 E

does not have a parent, we add a corresponding relation $(E_{\text{subj}}, E_{\text{obj}}, \Psi)$ with $E_{\text{obj}} = E$ and $\Psi = \text{parent_of}$.

E. 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. We extend the common hOCR format [34] to additionally accommodate hierarchical structures.

F. Implementation details

Our DSG system is based on the neural motifs architecture [31] using the implementations from [35], [32]. 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 ϕ_s^{\max} . If the longest side would exceed a predefined maximum size of ϕ_l^{\max} after this step, resizing is instead done so that the longest side length is ϕ_l^{\max} . During training, the image size is varied for augmentation purposes. For this, ϕ_s^{\max} is chosen randomly from a set of different sizes. We set ϕ_l^{\max} at 600, while ϕ_s^{\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 $\phi_s^{\max} = 400$ and $\phi_l^{\max} = 600$.

Entity detection (component C2): The entity detection component is based on the Faster R-CNN architecture [37].³ We use a ResNet [38] backbone of depth 50 in our system. Training of component C2 uses the loss term $L_{C2} = L_{\text{RPN}}^{\text{cls}} + L_{\text{RPN}}^{\text{loc}} + L_E^{\text{cls}} + L_E^{\text{loc}}$. The losses $L_{\text{RPN}}^{\text{loc}}$ and L_E^{cls} 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_E^{loc} and L_E^{cls} , 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 $\rho_1^{\text{rel_in}}$ are extracted via the underlying Faster R-CNN architecture for the bounding box B_j of each entity. Specifically, the convolutional neural network [38] 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) [39], 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

³We also experimented with a mask segmentation but could not measure a significant performance gain and, hence, discarded this subtask in our implementation.

alignment layer, concatenated, and passed through two fully-connected layers with ReLu activations [41] to produce feature vectors of dimension 1024. These feature vectors are then used as the relation context input feature $\rho_{\text{vis}}^{\text{rel_in}}$.

Following [31], we extract visual feature maps ρ_1^{pair} for the union bounding box B_{pair} of subject-object pairs $(E_{\text{subj}}, E_{\text{obj}})$. The extraction proceeds analogously to $\rho_{\text{vis}}^{\text{rel_in}}$ but uses the region B_{pair} to filter the multi-scale visual feature maps, resulting in pair-wise features ρ_1^{pair} . ρ_1^{pair} and ρ_2^{pair} are fed into fully-connected layers W_1^{pair} and W_2^{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 $W_{\text{rel}}^{\text{pair}}$ with an output dimension equal to the number of relation types and added to the frequency bias term ρ_3^{pair} , resulting in the pair-wise output feature $\rho_{\text{out}}^{\text{pair}} = W_{\text{rel}}^{\text{pair}}((W_1^{\text{pair}} \rho_1^{\text{pair}}) \circ (W_2^{\text{pair}} \rho_2^{\text{pair}})) + \rho_3^{\text{pair}}$. $\rho_{\text{out}}^{\text{pair}}$ 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. VI-A) scores $S_{\text{subj}} = \text{IoU}(E_{\text{subj}}, E_{\text{subj}}^{\text{GT}})$ and $S_{\text{obj}} = \text{IoU}(E_{\text{obj}}, E_{\text{obj}}^{\text{GT}})$ 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 $\rho_{\text{vis}}^{\text{ref_in}}$ is computed analogously to $\rho_{\text{vis}}^{\text{rel_in}}$. The category embedding $\rho_{\text{cat}}^{\text{ref_in}}$ is computed by mapping the name of the semantic category directly onto the GloVe word embedding with an identical name [40]. 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 \mapsto bibliography), (TEXT BLOCK \mapsto paragraph), (FIGURE CAPTION \mapsto caption), (FIGURE GRAPHIC \mapsto graphic), (PAGE NR. \mapsto numbering), (TABLE CAPTION \mapsto caption). The word embedding dimension is set to 200 in our experiments. For the positional embedding $\rho_{\text{pos}}^{\text{ref_in}}$, 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_{rel} and class refinement, L_{ref} .

V. DATASETS

We compare our system using two different datasets.⁴ 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 [11]. However, scientific articles follow a fairly similar structure. To

⁴A detailed overview is in our GitHub: <https://github.com/jds2cirwyc2w/DSG>

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.

A. arXivdocs-target

The existing dataset for training and evaluation, arXivdocs-target, contains 362 hand-annotated documents [11]. 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 [42] to allow processing with standardized benchmark libraries and document parsing codebases.

B. E-Periodica

E-Periodica is a project that aims to digitize a wide range of historical and contemporary magazines for future generations [43]. 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.⁵ 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 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., a single column with a separate article).

Further details and summary statistics are included in our GitHub.

VI. EXPERIMENTAL SETUP

A. 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 for our purpose of parsing document structures.

Entity detection: We follow common practice [42], [44] 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 [45]. Specifically, we calculate the so-called intersection-over-union (IoU) via

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

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 [44].

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

B. 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 [44] with copy-paste augmentation [46]. Second, since the COCO dataset does not contain documents, we then proceed to pre-train all systems on arXivdocs-weak [11]. 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.

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

⁵Examples are on GitHub.

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_{subj} and B_{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 ~ 0.1616 seconds. Integrating our grammar-based postprocessing comes with only a small overhead and results in an average joint processing time of ~ 0.1776 seconds for the same computational setup.

C. Baselines

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

- **DocParser** [11]: We reimplement DocParser [11], 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. II). To ensure comparability between DocParser and our system, we use the Faster R-CNN architecture with a ResNet [38] 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 [11] 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 (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 (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 (C2 frozen) and DSG 2-stage (C2 unfrozen), where the systems are 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., [31], [32]). 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_1^p, E_2^p, ϕ^p) is considered a match with a ground-truth triple (E_1^g, E_2^g, ϕ^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.

VII. RESULTS

A. 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 II (arXivdocs-target) and Table III (E-Periodica). Here, we report the results for our DSG (last rows). We further state the results for DocParser [11], 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 [11]. 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 I provides a breakdown by different semantic categories on the E-Periodica dataset. Evidently, our DSG is consistently better for the vast majority of semantic cate-

Semantic category	DocParser [11]		DSG (ours)	
	IoU=0.5	IoU=0.75	IoU=0.5	IoU=0.75
mAP	55.32	35.33	62.35	41.26
ARTICLE	66.51	48.69	75.85	59.43
AUTHOR	41.32	16.92	47.40	20.40
BACKGR._FIG.	53.69	34.66	69.45	47.72
COLUMN	45.36	23.44	70.00	18.87
TEXTBLOCK	78.03	66.86	80.28	68.21
DOC._ROOT	99.01	99.01	99.01	99.01
FIGURE	42.10	19.08	56.64	36.75
FIG._CAPTION	32.58	23.15	41.10	31.45
FIG._GRAPHIC	66.29	50.10	76.55	58.32
FOOTER	42.59	8.51	56.99	18.07
FOOTNOTE	57.97	45.83	66.86	59.12
HEADER	44.83	15.87	32.24	13.36
HEADING	57.61	26.07	65.17	35.42
ITEM	46.76	30.62	56.04	37.01
ITEMIZE	53.72	40.87	69.73	56.40
META	83.97	83.97	87.23	87.23
ORDEREDGROUP	65.51	42.02	70.38	49.47
PAGE_NR.	62.93	1.80	68.21	7.22
ROW	49.75	17.00	52.12	23.28
TABLE	52.87	39.38	61.26	48.61
TABLE_OF_CONT.	51.90	21.99	47.18	20.20
TABULAR	52.58	37.84	57.54	31.28
UNORDERED_GROUP	24.52	18.99	26.79	22.20

TABLE I: Performance of entity detection per category on E-Periodica. Reported: average precision (AP) per semantic category. Compared: DocParser and DSG end-to-end.

gories.⁶ 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.

System	Variant	IoU=0.5	IoU=0.75
DocParser [11]	—	77.70	58.62
DSG 2-stage (ours)	C2 frozen	71.03	50.48
DSG 2-stage (ours)	C2 unfrozen	77.48	57.40
DSG end-to-end (ours)	w/o postproc.	79.61	58.58
DSG end-to-end (ours)	w/ postproc.	79.61	58.58

TABLE II: Performance (mAP) of entity detection on arXivdocs-target.

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,

⁶A detailed breakdown by different semantic categories is provided in our GitHub at <https://github.com/jds2cirwyc2w/DSG>

System	Variant	IoU=0.5	IoU=0.75
DocParser	–	55.32	35.33
DSG 2-stage (ours)	C2 frozen	51.51	10.84
DSG 2stage (ours)	C2 unfrozen	54.41	36.22
DSG end-to-end (ours)	w/o postproc.	62.35	41.26
DSG end-to-end (ours)	w/ postproc.	<u>62.18</u>	<u>40.90</u>

TABLE III: Performance of entity detection on E-Periodica.

namely, arXivdocs-target (Table IV) and E-Periodica (Table V).

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 [11] 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 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 the entity recognition. Nevertheless, the performance of our system is highly effective in practice where the aim is to recover the overall document structure.

Qualitative assessment : We performed a qualitative assessment (see our GitHub at <https://github.com/jds2cirwyc2w/> DSG). Thereby, we demonstrate that we generate meaningful and effective document structures in practice.

VIII. 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

System	Variant	Precision	Recall	F1
DocParser [11]	–	0.6646	0.7687	0.7054
DSG 2-stage (ours)	C2 frozen	<u>0.7689</u>	0.7042	0.7223
DSG 2-stage (ours)	C2 unfrozen	<u>0.7378</u>	0.7560	<u>0.7378</u>
DSG end-to-end (ours)	w/o postproc.	0.7709	<u>0.7649</u>	0.7592
DSG end-to-end (ours)	w/ postproc.	0.6959	0.7590	0.7185

TABLE IV: Performance of structure parsing on arXivdocs-target.

System	Variant	Precision	Recall	F1
DocParser [11]	–	0.1725	0.2319	0.1884
DSG 2-stage (ours)	C2 frozen	0.3901	0.5083	0.4232
DSG 2-stage (ours)	C2 unfrozen	0.4589	<u>0.5276</u>	0.4740
DSG end-to-end (ours)	w/o postproc.	<u>0.5545</u>	0.5528	0.5340
DSG end-to-end (ours)	w/ postproc.	0.5701	0.5197	<u>0.5308</u>

TABLE V: Performance of structure parsing on E-Periodica.

are [4], [5], [6], [7], [8], [9]. Recent works (e.g., [18], [47]) 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., [48]) 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 [11] 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 [17] and without hierarchical information [49].

Conclusion: In this paper, 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.

REFERENCES

- [1] D. Johnson, “Pdf statistics—the universe of electronic documents,” 2018.
- [2] S. B. Johnson, D. A. Campbell, M. Krauthammer, P. K. Tulipano, E. A. Mendonça, C. Friedman, and G. Hripcsak, “A Native XML Database Design for Clinical Document Research,” *AMIA Annual Symposium Proceedings*, 2003.
- [3] C. Clifton, H. Garcia-Molina, and R. Hagmann, “The Design of a Document Database,” in *Conference on Document Processing Systems (DocProcess)*, 2000.
- [4] D. Che, K. Aberer, and M. T. Özsü, “Query optimization in XML structured-document databases,” *The VLDB Journal*, vol. 15, no. 3, pp. 263–289, 2006.
- [5] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang, “WebTables: Exploring the power of tables on the web,” *Proceedings of the VLDB Endowment*, vol. 1, no. 1, 2008.
- [6] R. Wilkinson, “Effective Retrieval of Structured Documents,” in *Conference on Research and Development in Information Retrieval (SIGIR)*, B. W. Croft and C. J. van Rijsbergen, Eds., 1994.
- [7] Q. Li and B. Moon, “Indexing and Querying XML Data for Regular Path Expressions,” in *International Conference on Very Large Data Bases (VLDB)*, 2001.
- [8] T. Manabe and K. Tajima, “Extracting logical hierarchical structure of HTML documents based on headings,” *Proceedings of the VLDB Endowment*, vol. 8, no. 12, pp. 1606–1617, 2015.
- [9] S. Wu, L. Hsiao, X. Cheng, B. Hancock, T. Rekatsinas, P. Levis, and C. Ré, “Fonduer: Knowledge Base Construction from Richly Formatted Data,” in *International Conference on Management of Data (SIGMOD)*, 2018.
- [10] G. M. Bimakhashen and S. A. Mahmoud, “Document layout analysis: A comprehensive survey,” *Acm Computing Surveys*, vol. 52, no. 6, Oct. 2019.
- [11] J. Rausch, O. Martinez, F. Bissig, C. Zhang, and S. Feuerriegel, “DocParser: Hierarchical Document Structure Parsing from Renderings,” in *Conference on Artificial Intelligence (AAAI)*, 2021.
- [12] T. M. Breuel, “High Performance Text Recognition Using a Hybrid Convolutional-LSTM Implementation,” in *International Conference on Document Analysis and Recognition (ICDAR)*, 2017.
- [13] W. Zhu, N. Sokhandan, G. Yang, S. Martin, and S. Sathyaranayana, “DocBed: A Multi-Stage OCR Solution for Documents with Complex Layouts,” in *Conference on Artificial Intelligence (AAAI)*, 2022.
- [14] M. Li, L. Cui, S. Huang, F. Wei, M. Zhou, and Z. Li, “TableBank: A Benchmark Dataset for Table Detection and Recognition,” in *Conference on Language Resources and Evaluation (LREC)*, 2020.
- [15] B. Smock, R. Pesala, and R. Abraham, “PubTables-1M: Towards comprehensive table extraction from unstructured documents,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [16] A. Antonacopoulos, D. Bridson, C. Papadopoulos, and S. Pletschacher, “A Realistic Dataset for Performance Evaluation of Document Layout Analysis,” in *International Conference on Document Analysis and Recognition (ICDAR)*, 2009.
- [17] X. Zhong, J. Tang, and A. Jimeno Yepes, “PubLayNet: Largest Dataset Ever for Document Layout Analysis,” in *International Conference on Document Analysis and Recognition (ICDAR)*, 2019.
- [18] S. Appalaraju, B. Jasani, B. U. Kota, Y. Xie, and R. Manmatha, “DocFormer: End-to-End Transformer for Document Understanding,” in *International Conference on Computer Vision (ICCV)*, 2021.
- [19] U. Schäfer, B. Kiefer, C. Spurk, J. Steffen, and R. Wang, “The ACL Anthology Searchbench,” in *Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, 2011.
- [20] U. Schäfer and B. Weitz, “Combining OCR Outputs for Logical Document Structure Markup: Technical Background to the ACL 2012 Contributed Task,” in *Proceedings of the ACL-2012 Special Workshop on Rediscovering 50 Years of Discoveries*, 2012, pp. 104–109.
- [21] B. Yıldız, K. Kaiser, and S. Miksch, “Pdf2table: A Method to Extract Table Information from PDF Files,” *Indian International Conference on Artificial Intelligence (IICAI)*, 2005.
- [22] Y. Wang, I. T. Phillips, and R. M. Haralick, “Table Structure Understanding and its Performance Evaluation,” *Pattern Recognition*, vol. 37, no. 7, pp. 1479–1497, 2004.
- [23] A. Gilani, S. R. Qasim, I. Malik, and F. Shafait, “Table Detection Using Deep Learning,” in *International Conference on Document Analysis and Recognition (ICDAR)*, 2017.
- [24] S. A. Siddiqui, M. I. Malik, S. Agne, A. Dengel, and S. Ahmed, “DeCNT: Deep deformable CNN for table detection,” *IEEE Access*, pp. 74 151–74 161, 2018.
- [25] T. Kieninger and A. Dengel, “The T-recs table recognition and analysis system,” in *International Workshop on Document Analysis Systems (DAS)*, 1998.
- [26] A. Pivk, P. Cimiano, Y. Sure, M. Gams, V. Rajković, and R. Studer, “Transforming arbitrary tables into logical form with TARTAR,” *Data and Knowledge Engineering*, vol. 60, no. 3, pp. 567–595, 2007.
- [27] S. Schreiber, S. Agne, I. Wolf, A. Dengel, and S. Ahmed, “DeepDeSRT: Deep Learning for Detection and Structure Recognition of Tables in Document Images,” in *International Conference on Document Analysis and Recognition (ICDAR)*, 2017.
- [28] S. Joseph and J. George, “A Review of Various Line Segmentation Techniques Used in Handwritten Character Recognition,” in *Information and Communication Technology for Competitive Strategies (ICTCS)*, A. Joshi, M. Mahmud, and R. G. Ragel, Eds., 2021.
- [29] J. Ma, J. Du, P. Hu, Z. Zhang, J. Zhang, H. Zhu, and C. Liu, “HRDoc: Dataset and Baseline Method Toward Hierarchical Reconstruction of Document Structures,” in *Conference on Artificial Intelligence (AAAI)*, 2023.
- [30] X. Chang, P. Ren, P. Xu, Z. Li, X. Chen, and A. Hauptmann, “A Comprehensive Survey of Scene Graphs: Generation and Application,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 1, pp. 1–26, 2023.
- [31] R. Zellers, M. Yatskar, S. Thomson, and Y. Choi, “Neural Motifs: Scene Graph Parsing with Global Context,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [32] S. Khandelwal, M. Suhail, and L. Sigal, “Segmentation-Grounded Scene Graph Generation,” in *International Conference on Computer Vision (ICCV)*, 2021.
- [33] K. Tang, H. Zhang, B. Wu, W. Luo, and W. Liu, “Learning to Compose Dynamic Tree Structures for Visual Contexts,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [34] T. Breuel, “The hOCR Microformat for OCR Workflow and Results,” in *International Conference on Document Analysis and Recognition (ICDAR)*, 2007.
- [35] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, “Detectron2,” <https://github.com/facebookresearch/detectron2>, 2019.
- [36] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.
- [37] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.
- [38] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [39] T. Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature pyramid networks for object detection,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [40] J. Pennington, R. Socher, and C. Manning, “Glove: Global Vectors for Word Representation,” in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.
- [41] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *International Conference on Machine Learning (ICML)*, 2010.
- [42] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, M. S. Bernstein, and L. Fei-Fei, “Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations,” *International Journal of Computer Vision*, vol. 123, no. 1, pp. 32–73, 2017.

- [43] R. Wanger, “E-Periodica: die Plattform für digitalisierte Schweizer Zeitschriften,” in *Bibliotheken der Schweiz: Innovation durch Kooperation*, Zentralbibliothek Zürich, A. Keller, and S. Uhl, Eds. De Gruyter, Jun. 2018, pp. 401–413.
- [44] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft COCO: Common Objects in Context,” in *European Conference on Computer Vision (ECCV)*, 2014.
- [45] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, A. Zisserman, M. Everingham, L. V. Gool, K. Leuten, B. C. K. I. Williams, J. Winn, and A. Zisserman, “The PASCAL Visual Object Classes (VOC) Challenge,” *International Journal of Computer Vision*, vol. 88, 2009.
- [46] G. Ghiasi, Y. Cui, A. Srinivas, R. Qian, T.-Y. Lin, E. D. Cubuk, Q. V. Le, and B. Zoph, “Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [47] Y. Huang, T. Lv, L. Cui, Y. Lu, and F. Wei, “LayoutLMv3: Pre-training for Document AI with Unified Text and Image Masking,” in *International Conference on Multimedia (MM)*, 2022.
- [48] L. Li, F. Gao, J. Bu, Y. Wang, Z. Yu, and Q. Zheng, “An End-to-End OCR Text Re-organization Sequence Learning for Rich-Text Detail Image Comprehension,” in *European Conference on Computer Vision (ECCV)*, 2020.
- [49] M. Li, Y. Xu, L. Cui, S. Huang, F. Wei, Z. Li, and M. Zhou, “DocBank: A Benchmark Dataset for Document Layout Analysis,” in *International Conference on Computational Linguistics (COLING)*, 2020.
- [50] J. Gu, J. Kuen, V. I. Morariu, H. Zhao, R. Jain, N. Barmpaliasos, A. Nenkova, and T. Sun, “UniDoc: Unified Pretraining Framework for Document Understanding,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
- [51] J. Li, Y. Xu, T. Lv, L. Cui, C. Zhang, and F. Wei, “DiT: Self-supervised Pre-training for Document Image Transformer,” in *International Conference on Multimedia (MM)*, 2022.
- [52] ABBYY, “ABBYY FineReader,” <https://pdf.abbyy.com/>, Mar. 2023.
- [53] Adobe, “Adobe Acrobat Pro,” <https://www.adobe.com/acrobat.html>, 2023.

APPENDIX A RELATED WORK

Table VI shows an overview of key systems for document structure parsing.

System	Document entities	Hierarchical structures	End-to-end training
PubLayNet [17]	✓	✗	✓
DocFormer [18]	✓	✗	✓
UniDoc [50]	✓	✗	✓
DiT [51]	✓	✗	✓
HRDoc [29]	only line-level	✓	✓
TableBank [14]	only tables	✓	✓
PubTables [15]	only tables	✓	✗
DocParser [11]	✓	✓	✗
DSG (<i>ours</i>)	✓	✓	✓

TABLE VI: Overview of key systems for document structure parsing from document renderings.

APPENDIX B OUR DSG SYSTEM

A. Grammar-based postprocessing (C4)

g_{ilg} 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 $\Psi \in \{\text{parent_of}, \text{followed_by}\}$ 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 $\Psi \in \{\text{followed_by}, \text{parent_of}\}$ 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 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_{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 $(E_{\text{cand}}, E_{\text{missing_parent}}, \Psi)$ with $\Psi = \text{parent_of}$ and candidate parent entities E_{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_{ilg} .

B. hOCR conversion engine (C5)

The hOCR format [34] 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 map a 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 η . We provide a list of (c, η) tuples in Table VII. 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.

In order to deal with both semantic categories, we keep the DSG-specific XML nodes to preserve the underlying structural information. We provide a list of (c, η) tuples in Table VII.

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 $H'_{\text{parent_of}}$ that only considers relations of type $\Psi = \text{parent_of}$. For this, hOCR and additional DSG XML nodes are initialized according to $H'_{\text{parent_of}}$ and the category mapping ω .
- 2) **Order of children** (s_2): 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_{leaf} in the document structure. Second, words are assigned to the entity E_{leaf} with the highest intersection-over-union score (see Sec. VI-A) between entity bounding box B_{leaf} and word bounding box B_{word} .

A key feature of the hOCR format is the ability to perform structure-based XPath queries on document files [34]. 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.

DSG semantic cat.	hOCR class	DSG semantic cat	hOCR class
DOC. ROOT	ocr_page	ITEM	ocr_carea
META	None	ITEMIZE	ocr_float
AUTHOR	ocr_author	ORDERED GROUP	ocr_carea
BACKGROUND FIG.	ocr_float	PAGE NR.	ocr_pageno
TEXT BLOCK	ocrx_block	TABLE	ocr_table
FIGURE	ocr_float	TABULAR	ocr_table
FIGURE GRAPHIC	ocr_photo	TABLE OF CONTENTS	ocr_table
FIGURE CAPTION	ocr_caption	UNORDERED GROUP	ocr_float
FOOTER	ocr_footer	ARTICLE	None
FOOTNOTE	ocr_footer	COLUMN	ocr_carea
HEADER	ocr_header	ROW	ocr_carea
HEADING	ocr_header		

TABLE VII: Mapping between semantic categories in DSG and hOCR classes.

- 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.

APPENDIX C DATASETS

Dataset	Document type	#Docs	#Categories	Source
arXivdocs-target	Scientific articles	362	21	[11]
E-Periodica	Magazines	542	22	Ours

TABLE VIII: Overview of existing datasets for our task.

An overview of the two datasets used in our experiments is given in Table VIII.

A. E-Periodica

Figure 4 shows the distribution of page types in the E-Periodica dataset.

Summary statistics: Table IX 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 [11], implying that E-Periodica is a challenging dataset for benchmarking.

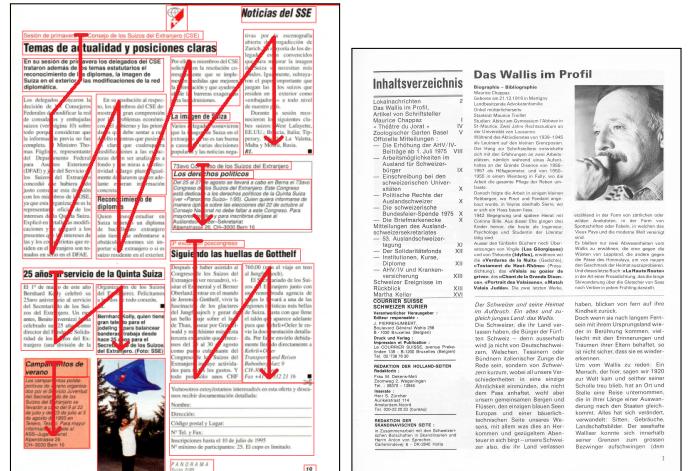
The distribution of leaf nodes per document in Fig. 3c further emphasizes the complexity of our dataset due to the deeply nested structure.

APPENDIX D RESULTS

A. 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** [52] and **Adobe Acrobat** [53]. These OCR



(a) Example of a complex reading (b) Example of document with with a table of contents and an article.

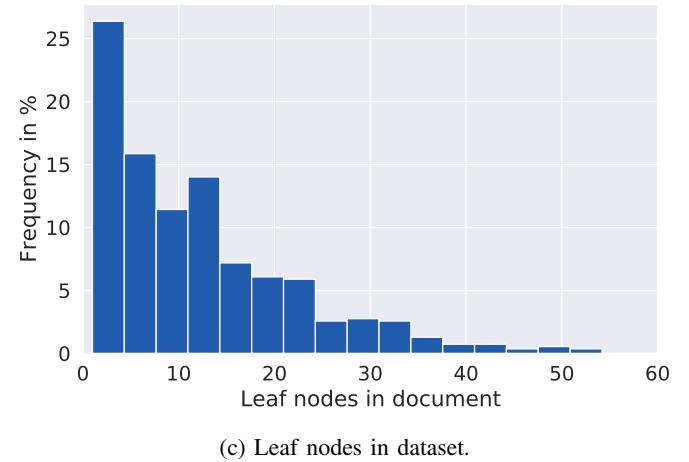


Fig. 3: Examples and statistics of documents featured in E-Periodica.

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

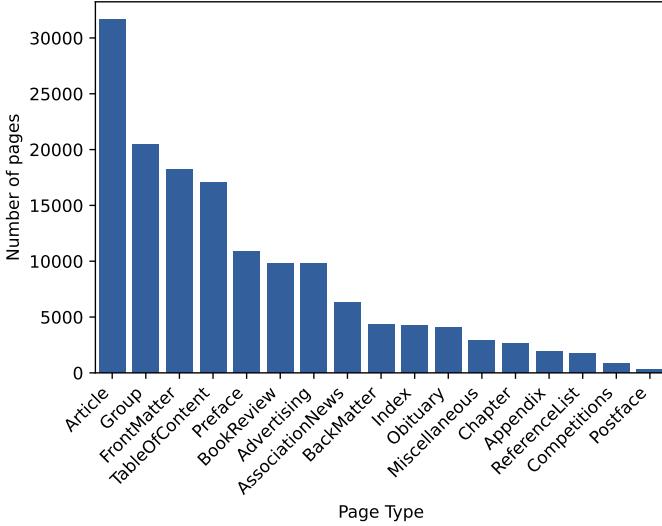


Fig. 4: Distribution of page types in the E-Periodica dataset.

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

TABLE IX: Distribution of semantic categories in E-Periodica

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 5 shows a representative example. The unedited input images are shown in Figure 3b 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.

APPENDIX E QUERYING

A. 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.

B. Examples

We demonstrate an exemplary query on real data (Fig. 5j) as follows:

```
>>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", "21.12.191", "6", "in", "Martigny"]
```

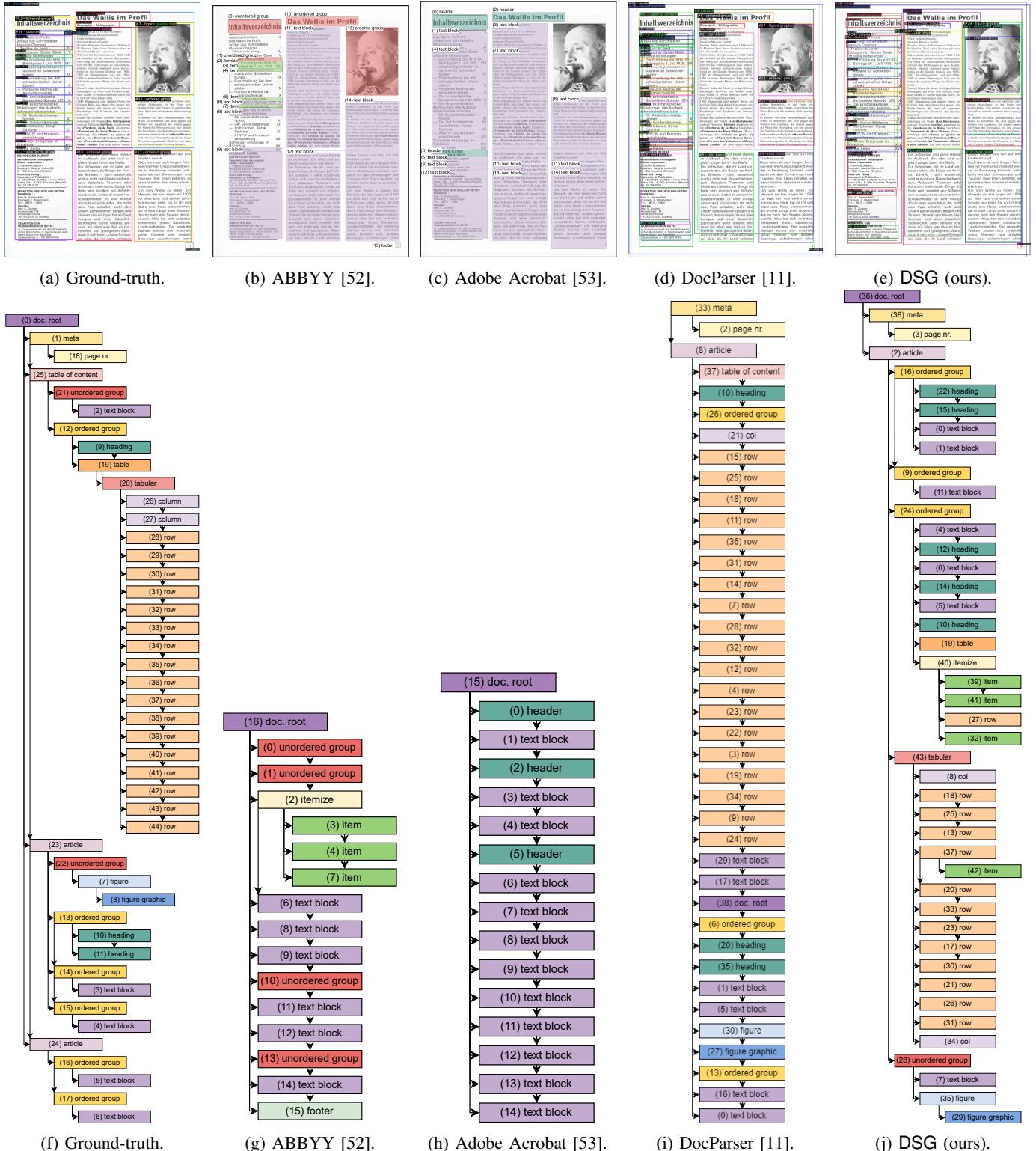


Fig. 5: Qualitative evaluation comparing the parsed hierarchical document structure by different systems. The document is characterized by complex, hierarchical structure. Top: entity recognition; bottom: hierarchical structure.