DocParser: A System for Leveraging Structured Documents in Knowledge Base Construction

Johannes Rausch
ETH Zurich
Zurich, Switzerland
johannes.rausch@inf.ethz.ch

Shuai Zhang
ETH Zurich
Zurich, Switzerland
shuazhang@inf.ethz.ch

David Dao
ETH Zurich
Zurich, Switzerland
david.dao@inf.ethz.ch

Stefan Feuerriegel
ETH Zurich
Zurich, Switzerland
sfeuerriegel@ethz.ch

Wentao Wu
Microsoft Research
Redmond, WA, USA
wentao.wu@microsoft.com

Ce Zhang
ETH Zurich
Zurich, Switzerland
ce.zhang@inf.ethz.ch

ABSTRACT

One major challenge in knowledge base construction (KBC) is the lack of document structure (e.g, in PDF or other scanned formats). Engineers therefore have to write ad hoc code to prepare documents for data analysis due to varying layouts and appearances. In this paper, we demonstrate DocParser, a system that automatically generates full hierarchical document representations from unstructured inputs. DocParser operates on document renderings and is agnostic to the underlying source file format. We showcase how DocParser facilitates the management and annotation of hierarchical document representations via a labeling GUI and scalable weak-supervision learning techniques. We also showcase how easy it is to integrate DocParser with downstream KBC pipelines.

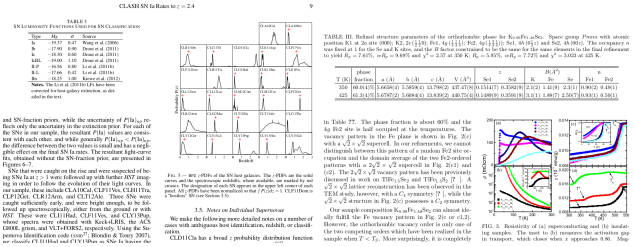
PVLDB Reference Format:
Johannes Rausch, Shuai Zhang, David Dao, Stefan Feuerriegel, Wentao Wu, Ce Zhang. DocParser: A System for Leveraging Structured Documents in Knowledge Base Construction. *PVLDB*, 13(xxx): xxxx-yyyy, 2020.
DOI: https://doi.org/10.14778/xxxxxxx.xxxxxxx

1. INTRODUCTION

Knowledge base construction (KBC), the process of populating a structured database from unstructured documents, is an indispensable component of modern data ecosystems. Over the last decade, KBC has received intensive interests from both database [4, 11, 16, 17, 19] and natural language processing communities [3, 5, 9, 10].

From our experience of building a diverse range of KBC applications, we find that the most time-consuming experience, surprisingly, is not the development of the extraction module itself. Instead, we see ourselves spending significant time and efforts writing *ad hoc* code to bring input documents into a form that downstream information extraction systems can process. Different applications contain documents with different layouts and styles, but *can we automate and unify this process with a single system that is robust across different applications?*

Proceedings of the VLDB Endowment, Vol. 13, No. xxx
ISSN 2150-8097.
DOI: https://doi.org/10.14778/xxxxxxx.xxxxxxx



(a) (b)

Figure 1: Example document snippets with diverse layouts and styles in our applications. The goal of DocParser is to translate these documents into a unified representation for downstream information extraction engines to process.

We demonstrate DocParser, a system that produces an intermediate representation for a diverse range of documents. DocParser takes as input a PDF document or document rendering, and translates it into a JSON file containing the natural semantic hierarchy that represents a document (e.g., title, abstract, section header, text block, figure, caption, table, table rows, columns, cells, etc.). These JSON files can be managed and queried using a document database, and can also be used by downstream information extraction engines such as DeepDive [19], Snorkel [12], and Fonduer [18].

DocParser treats the task of parsing hierarchical structures from documents as a *computer vision* problem. As a result, it is robust to document formats (e.g., `.pdf`, `.docx`). We identify three main challenges in managing hierarchically structured documents:

1. The granularity and quantity of document entities (e. g., hundreds of entities such as table cells, text lines, etc.) can vastly exceed typical settings of computer vision tasks. Moreover, entities are not arranged in a flat list, but are organized into a tree-like structure.

2. The labeling complexity of hierarchical document annotation is extremely high. Manual annotation of entities and relations can take over 10 minutes for a single document page.

3. Development of downstream KBC pipelines is hindered by the lack of access to structural information in unstructured documents (e. g., PDF or scanned documents).

Summary of Demonstration Plan: We plan to demonstrate how DocParser addresses these challenges via the following scenarios:

- A demonstration of the labeling interface of DocParser and how it handles annotation and display of hundreds of hierarchically nested elements on a single page.
- A demonstration of how DocParser reduces labeling complexity by a factor of 8 for full document hierarchy parsing through weak supervision. Instead of asking human experts to annotate large sets of training examples, we model the document structural parsing problem as the *inverse process* of a LaTeX compiler.
- A demonstration on how to use the hierarchical document structures produced by DocParser for downstream KBC tasks.

2. DOCPARSER SYSTEM

2.1 System Overview

As presented by Figure 2, DocParser operates in two stages: (1) *detection* of document entities and (2) *classification* of hierarchical relations between entities. In order to reduce the high labeling complexity, our system generates noisy labeled training data from structured source files as additional weak supervision signals. For more details of the system design, we refer the readers to [14].

Detection of Document Entities: We employ a convolutional neural network, Mask R-CNN [8], for detection of document entities in our system. Formally, our detector takes as input a document rendering D_i and outputs a flat list of document entities $E_1, \ldots, E_m$. Each entity is described by its semantic category c_j, its rectangular bounding box region B_j on the document rendering, and a confidence score P_j.

Classification of Hierarchical Relations: To produce a full document graph from the detected entities, we classify all relation triplets $(E_{\text{subj}}, E_{\text{obj}}, \Psi)$ in the document. We use three relation types $\Psi \in \{is\_parent\_of, is\_followed\_by, \text{null}\}$, which enables reconstruction of the full hierarchical structure and allows to account for entities with meta-information that has no designated order (e. g., headers, page numbers, etc.). We use a set of heuristics to analyze properties such as visual nesting or left-right/top-bottom arrangement to assign relation types to all candidate triplets. Here, a user-defined document grammar provides a set of rules that have to be followed by the output document graph, e. g., by only allowing nesting relations of "table caption" entities with other entities belonging to the category "table." We refer readers to [14] for a detailed explanation of all applied heuristics.

Scalable Weak Supervision: Manual annotation of document entities yields labels of high quality, but is very costly and time-consuming. Training a system that can detect the full document hierarchy requires annotation of low-level entities such as table cells, section headers, nested figures, etc. As a result, labeling complexity can become a major obstacle for training robust systems. To reduce the need for manual labels, our system employs a scalable weak-supervision component. Our weak supervision uses an additional dataset of source codes that were used to generate document renderings. We generate a noisy mapping between document renderings and semantic entities/relations in the source codes. We observe significant performance gains when integrating weak supervision into our system.

2.2 User Interface

Document Inspection and Annotation: DocParser provides a user interface that allows users to import their own datasets and manage full hierarchical annotations for documents (see Figure 3). Documents can be imported into our system in PDF files or in image format. DocParser automatically generates JSON files that specify meta information and basic document structure.

After importing, individual documents can be annotated manually. The labeling view consists of two sections. The first section displays a document page as well as all annotated document entities. The second section displays the associated document tree that describes all inter-entity relations such as *nesting* and *reading order*. High-level structure entities such as figures and tables are not labeled explicitly. Instead, their bounding boxes are dynamically generated from the union of their child entities. This greatly facilitates the inspection and annotation process, as documents can contain hundreds of hierarchically nested entities (e. g., text blocks, text lines, tables, table cells, etc.) and only leaf nodes need explicit bounding box annotation.

Weak Supervision and Fine-Tuning: By providing our system with weak-supervision sources, users can automatically generate weak-supervision examples for their own datasets. Using our document annotation GUI, users can manually label a small number of documents in the target dataset. These documents are used to fine-tune DocParser in the target domain.

3. DEMONSTRATION SCENARIOS

We demonstrate three scenarios that showcase how DocParser can benefit users in handling hierarchically structured documents. We plan to share source codes for all scenarios, including the labeling GUI and library for document parsing, in the near future.

3.1 Hierarchical Document Annotation GUI

Hierarchical Document Annotation: We demonstrate the effectiveness of our GUI for annotating relations, e. g., hierarchies in tables, nested figures, or general reading order. Users can either use sample documents from our dataset of scientific articles, or import any custom documents in PDF or image format. Naive assignment of relation categories to entity pairs would be very time-consuming and lack visual feedback regarding the validity of assigned labels. Instead, we provide a dynamic tree-view of the document graph (see Figure 3). Individual nodes in the tree can be rearranged flexibly. The tree structure is automatically converted into a set of relation triplets for processing by DocParser.

The tree- and document-views are visually linked. When hovering the cursor over high-level structural elements in the tree, e. g., tables or figures, all of their corresponding child entities are highlighted in the document view. This further aids intuitive management of complex annotation structures that exist in real-world documents. We additionally demonstrate number of features that are based around particularly time-consuming tasks such as table annotation. For this purpose, we implement functions to automatically group table cells into a joint tabular structure or assign table and row enumerations.

Automatic Labeling and Initialization: In addition to manual labeling, we showcase how our system can further facilitate the labeling process. We employ our pre-trained models or simple heuristics to automatically generate document graphs for input documents (see Figure 4). Automatically labeled documents can be used as a basis for manual annotation. As a result, manual annotation can be significantly sped up by having human labelers focus on only a small fraction of corrections.

3.2 Demonstration of Weak Supervision

We demonstrate the effectiveness of our scalable weak supervision component to the audience with two scenarios.

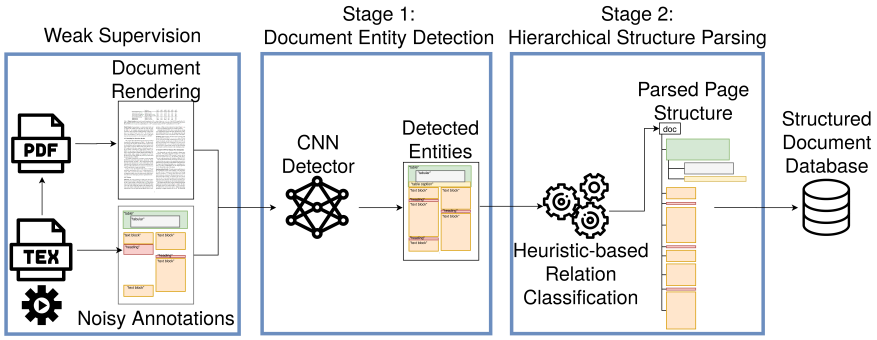


Figure 2: System overview. Document renderings are processed in two stages: (1) detection of document entities, e. g., tables, text blocks, etc. (2) classification of hierarchical relations between document entities, e. g., *is_parent_of*, *is_followed_by*. LaTeX source files are used in our weak supervision to increase performance and reduce manual labeling complexity.

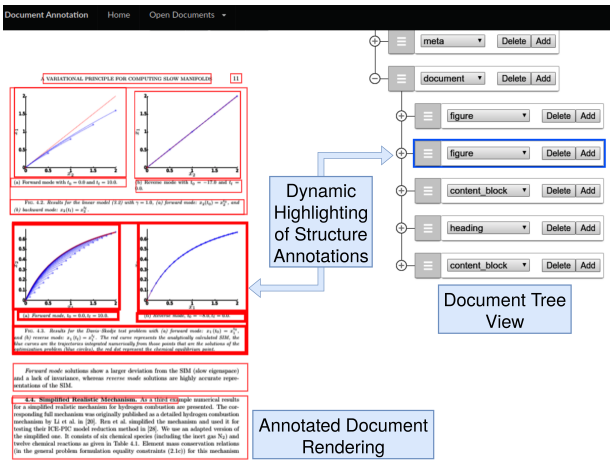


Figure 3: Inspection and annotation GUI: Users can manually annotate documents by adding annotations that consist of a bounding box on the document rendering and structure information (e.g., class, hierarchical relations) in a document graph view on the right. Selecting different nodes in the graph dynamically highlights all associated child entities in the document to facilitate inspection of the highly structured data (e.g., the "table" annotation here with bold red borders).

Datasets with source codes: Here we apply our scalable weak supervision to generate training signals that match the target domain, e. g., scientific articles. We compare the outputs of three variants of our system: (1) variants that have been trained on all available manual labels ("Baseline"), (2) variants trained purely via weak supervision on our generated examples ("WS"), and (3) variants generated by adding fine-tuning on varying amounts of manual examples in addition to weakly supervised training ("WS-FT"). We highlight that the third variant ("WS-FT") significantly outperforms the baseline, while using only a small subset of manual labels (see Figure 5). Using our GUI, the audience can interactively inspect and compare individual output document graphs (see Figure 6).

```
entity_detector = stage1_entity_detector.EntityDetector()
structure_parser = stage2_structure_parser.StructureParser()
entity_detector.init_model(default_weights='highlevel_wsft')
sample_img = skimage.io.imread(demo_img_path)
entity_predictions = entity_detector.predict(sample_img)
struct_dict = structure_parser.create_structure_for_doc(entity_predictions)
structure_parser.create_gui_doc_entry(gui_root_folder, struct_dict,
                                      demo_img_path, out_tag='wsft')
```

Figure 4: Automatic annotation of new documents in the database with DocParser.

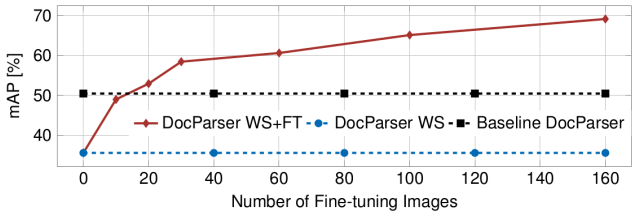
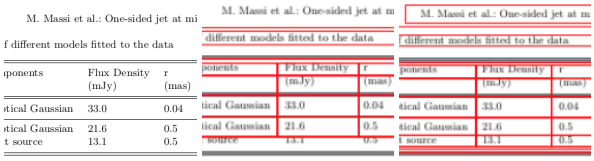


Figure 5: Weak supervision (WS), coupled with fine-tuning (WS+FT), surpasses manual baseline systems for entity detection while requiring 8x fewer hand-labeled images [14].



(a) (b) (c)

Figure 6: Output structure of our system for an input rendering region (a). We compare the weak supervision variant (b) with the variant (c) that has been additionally fine-tuned. Our weak supervision does not include a subset of classes, e. g. document headers. As a result, variant (b) fails to correctly identify the header located above the table. Our fine-tuned variant (c) effectively compensates such mismatches.

Datasets without source codes: We show that our weak supervision yields significant performance improvements, even when applied to new target domains. To demonstrate this, we apply DocParser on a benchmark dataset for table structure detection [7]. We observe that the pre-trained system using weak supervision is able to detect many table structures, despite the mismatch between source and target domains. By further fine-tuning the pre-trained system, we can significantly outperform the manually trained baseline and achieve state-of-the-art results for the benchmark task. In our demonstration, we show how to use DocParser to automatically extract structures on this new dataset.

3.3 Use in Downstream KBC Tasks

Our system produces full hierarchical document structures that can be used in downstream KBC tasks. Popular analysis pipelines apply data programming to generate large training sets for relation extraction [12, 13]. Furthermore, incorporating document structure can improve performance of such extraction systems [18]. We show how to integrate the parsed document structure in such KBC pipelines by defining additional labeling functions (see Figure 7).

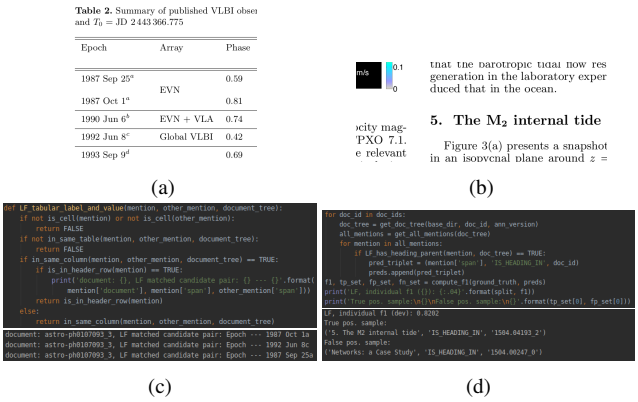


Figure 7: Structure information is retrieved from input document renderings (a) and (b) to enrich mentions with document structure information. This information allows definition of simple, but powerful extraction rules such as (c) and (d).

4. RELATED WORK

Structure Parsing in Document Renderings: The use of document renderings as inputs for related tasks such as table detection and table structure parsing has been extensively studied. Recent approaches apply convolutional neural networks (CNNs) for the detection of table entities [2, 6] or classification of row and column pixels [15]. Table structures, however, are not sufficient to enclose the content and structure of most real-world documents. Approaches for layout analysis and page segmentation consider full documents and larger sets of entity types, but do not generate full hierarchical representations [1]. We thus view our system as an important step towards making the full structure of documents accessible to downstream KBC tasks.

Weak Supervision in KBC: Weak supervision through data programming is an effective technique in KBC [12, 13, 18]. Instead of manual labeling, a set of noisy labeling functions are defined to train a noise-aware generative model to enable automatic labeling of large training sets. In contrast, DocParser deals with noise introduced by its weak supervision through fine-tuning with a small number of manual examples. We view KBC systems that employ data programming as a particularly suited counterpart in holistic information extraction systems: They can leverage the additional structural information in the KBC workflow, and are able to deal with potential noise that is introduced in the automatic structure parsing process with DocParser.

5. CONCLUSION

We have demonstrated DocParser, a system that enables efficient parsing of full document structures using scalable weak supervision. Our work can help researchers and developers build more robust and effective KBC systems by providing a unified document representation for downstream analysis applications. As future work, we plan to extend and test our system on a larger variety of document types, by incorporating weak supervision sources from more domains that are commonplace in many real-world workflows, such as office file formats.

6. REFERENCES

[1] A. Antonacopoulos, S. Pletschacher, D. Bridson, and C. Papadopoulos. ICDAR2009 page segmentation competition. In *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, 2009.

[2] S. Arif and F. Shafait. Table Detection in Document Images using Foreground and Background Features. In *2018 Digital Image Computing: Techniques and Applications (DICTA)*. IEEE, 2019.

[3] J. Betteridge, A. Carlson, S. A. Hong, E. R. Hruschka, E. L. Law, T. M.mitchell, and S. H. Wang. Toward never ending language learning. In *AAAI Spring Symposium - Technical Report*, volume SS-09-07, 2009.

[4] Y. Chen and D. Z. Wang. Knowledge expansion over probabilistic knowledge bases. In *SIGMOD*, 2014.

[5] X. L. Dong, E. Gabrilovich, G. Heitz, W. Horn, K. Murphy, S. Sun, and W. Zhang. From data fusion to knowledge fusion. *Proceedings of the VLDB Endowment*, 7(10), 2014.

[6] A. Gilani, S. R. Qasim, I. Malik, and F. Shafait. Table Detection Using Deep Learning. *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, 1, 2017.

[7] M. Gobel, T. Hassan, E. Oro, and G. Orsi. ICDAR 2013 table competition. In *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*. IEEE, 2013.

[8] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *Proceedings of the IEEE international conference on computer vision*, 2017.

[9] S. Jiang, D. Lowd, and D. Dou. Learning to refine an automatically extracted knowledge base using Markov logic. In *Proceedings - IEEE International Conference on Data Mining, ICDM*, 2012.

[10] Y. Li, F. R. Reiss, and L. Chiticariu. SystemT: A declarative Information Extraction system. In *ACL*, 2011.

[11] N. Nakashole, M. Theobald, and G. Weikum. Scalable knowledge harvesting with high precision and high recall. In *WSDM*, New York, New York, USA, 2011. ACM Press.

[12] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré. Snorkel: Rapid Training Data Creation with Weak Supervision. *Proceedings of the VLDB Endowment*, 11(3), 2017.

[13] A. Ratner, C. De Sa, S. Wu, D. Selsam, and C. Ré. Data Programming: Creating Large Training Sets, Quickly. In *Advances in neural information processing systems*, 2016.

[14] J. Rausch, O. Martinez, F. Bissig, C. Zhang, and S. Feuerriegel. DocParser: Hierarchical Structure Parsing of Document Renderings, 2019. arXiv:1911.01702.

[15] 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 *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, volume 1. IEEE, 2018.

[16] W. Shen, A. H. Doan, J. F. Naughton, and R. Ramakrishnan. Declarative information extraction using datalog with embedded extraction predicates. In *VLDB*, 2007.

[17] G. Weikum and M. Theobald. From information to knowledge: Harvesting entities and relationships from web sources. In *PODS*, 2010.

[18] S. Wu, L. Hsiao, X. Cheng, B. Hancock, T. Rekatsinas, P. Levis, and C. Ré. Fonduer: Knowledge Base Construction from Richly Formatted Data, 2017. arXiv:1703.05028.

[19] C. Zhang, C. Ré, M. Cafarella, C. De Sa, A. Ratner, J. Shin, F. Wang, and S. Wu. DeepDive: Declarative knowledge base construction. *Communications of the ACM*, 60(5), 2017.