# DocMaster: A Unified Platform for Annotation, Training, & Inference in Document Question-Answering

**Alex Nguyen**[◇] **Zilong Wang**[◇] **Jingbo Shang**[◇,♡,♠] **Dheeraj Mekala**[◇,♠]
[◇]University of California San Diego
[♡] Halıcıoğlu Data Science Institute, University of California San Diego
{atn021, zlwang, jshang, dmekala}@ucsd.edu

## Abstract

The application of natural language processing models to PDF documents is pivotal for various business applications yet the challenge of training models for this purpose persists in businesses due to specific hurdles. These include the complexity of working with PDF formats that necessitate parsing text and layout information for curating training data and the lack of privacy-preserving annotation tools. This paper introduces DocMaster, a unified platform designed for annotating PDF documents, model training, and inference, tailored to document question-answering. The annotation interface enables users to input questions and highlight text spans within the PDF file as answers, saving layout information and text spans accordingly. Furthermore, DocMaster supports both state-of-the-art layout-aware and text models for comprehensive training purposes. Importantly, as annotations, training, and inference occur on-device, it also safeguards privacy. The platform has been instrumental in driving several research prototypes concerning document analysis such as the AI assistant utilized by University of California San Diego's (UCSD) International Services and Engagement Office (ISEO) for processing a substantial volume of PDF documents.

## 1 Introduction

Documents and forms are omnipresent within enterprises encompassing financial bills like invoices, purchase records, financial statements, and official communications such as notices and announcements. The application of machine learning for automating document processing stands to significantly accelerate processing times (Tan et al., 2023).

Visually rich document understanding has recently attracted much attention from researchers. A simple approach involves parsing text from PDFs and leveraging established Natural Language Processing (NLP) models (Devlin et al., 2019; Liu et al., 2019; Brown et al., 2020). However, these methods overlook the valuable layout information embedded within PDFs. A series of works have been done to incorporate the layout features into the pre-training framework. LayoutLM (Xu et al., 2020) first proposes to encode the spatial relationships of words by embedding their position coordinates in an embedding layer. Following this direction, Xu et al. (2021); Huang et al. (2022) move beyond basic embedding techniques and specifically adapt the attention layers of the Transformer architecture to model the relative positional relationships within the 2D space of document pages. Gu et al. (2021); Wang et al. (2022), on the other hand, purse a more comprehensive understanding of the layout structure. They achieve this by encoding the hierarchical relation in the documents.

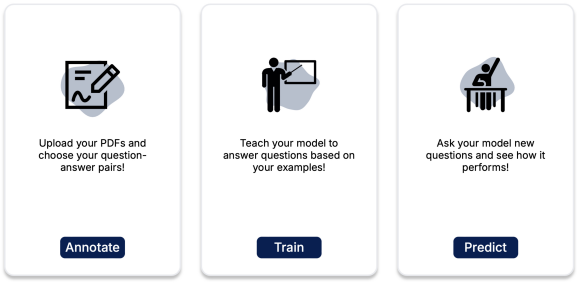



Figure 1: DocMaster supports annotation, model training, and inference functionalities for document question-answering in a single platform.

Despite the availability of these models, the persistent challenge lies in training them with custom business data due to particular obstacles. Firstly, working with the intricacies of the PDF format proves to be a nontrivial task (Lo et al., 2023). PDFs store text as character glyphs along with their positions on a page, necessitating complex operations to convert this data into usable text for NLP

♠ Corresponding Authors



*Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: System Demonstrations)*, pages 128–136

models. Operations like inferring token boundaries and managing white spacing are error-prone and add to the complexity. Secondly, organizations frequently handle sensitive documents that demand in-house tools for annotating and curating training data.

Addressing these obstacles, we introduce DocMaster, a unified platform designed for annotating PDF documents, model training, and inference for the question-answering (QA) task, as shown in Figure 1. The annotation interface is designed to maintain the layout integrity, requiring users to upload PDFs, provide questions, and highlight their specific text spans as answers in the PDFs. Once identified, it processes the PDF content, saving both textual and layout details. Privacy measures involve on-device processing, eliminating reliance on third-party services, and securely storing annotations within a local database. DocMaster accommodates an extensive array of models, encompassing both layout-aware models like LayoutLM (Xu et al., 2020) and text-only ones such as RoBERTa (Liu et al., 2019). The inference interface is user-friendly and accepts a PDF document and a trained model. It simplifies the task of locating answers to specific questions by highlighting relevant spans within the PDF document.

We deployed DocMaster in a practical scenario within the ISEO at UCSD, addressing the processing of hundreds of supporting documents for students to issue their work permits. Previously, staff members engaged in the manual review of each document before approving work permits. Through DocMaster, they annotated and trained a QA model seamlessly, leading to a remarkable seven-fold increase in the average number of documents processed per hour.

We present video demonstration, live demo website, and code on the project webpage.[1].

## 2 Related Work

**Language Modeling with PDFs** PDFs are widely used in daily life. It is trivial to resort to traditional language models, such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), and T5 (Raffel et al., 2020), to automatically understand the document contents. However, unlike the pure-text documents (Mekala et al., 2022a), PDFs carry rich information not only through the textual contents but also via the rich layout structure, presenting challenges for language models to comprehensively understand their contents. Xu et al. (2020); Hong et al. (2022); Garncarek et al. (2021) propose to use the coordinates of words in the page as the representation for the layout structure. They embed the coordinates in the embedding layer and add relative weights in the self-attention layers. Xu et al. (2021); Huang et al. (2022) incorporate the visual features from the document images. Following the previous works, Tang et al. (2023); Lv et al. (2023); Perot et al. (2023) enlarge the scale of pretraining and improve language models capability in understanding PDFs of various formats.

**Systems for Document AI** Document AI is drawing significant interest from both academia and industry. In addition to various language modeling techniques, major companies have also launched their proprietary Document AI services, including Google Cloud [2], Microsoft Azure [3], Amazon Web Services [4], etc. Although proprietary systems offer convenient and stable services, they are primarily business-oriented and lack transparency for those outside the company. Additionally, there are non-commercial Document AI systems available, such as Lo et al. (2023); Bryan et al. (2023). However, none of these systems comprehensively enable users to combine annotation, training, and inference within a single system. In contrast, DocMaster allows users to navigate the entire pipeline of Document-QA task, successfully eliminating programming barriers that hinder general users from utilizing Document AI tools.

## 3 DocMaster: Design

This section delves into the design aspects of our platform. DocMaster has three interfaces: (1) The Annotation interface, which processes a zip file containing PDF documents, enabling user annotation through text highlighting. (2) The Training interface, facilitating the training of both layout-aware and text models. (3) The Inference interface, which accepts a set of documents as input, allows users to select their trained model and highlights predictions on the PDFs. The DocMaster application is intended to run on the organization's

[1] https://alextongdo.github.io/doc-master-webpage/

[2] https://cloud.google.com/document-ai

[3] https://azure.microsoft.com/en-us/products/ai-services/ai-document-intelligence

[4] https://aws.amazon.com/machine-learning/ml-use-cases/document-processing/fintech/

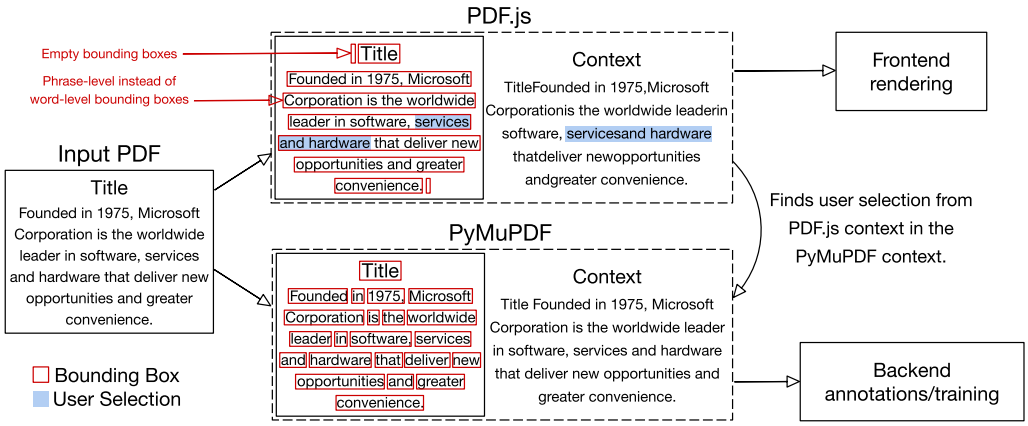



Figure 2: Training and inference with layout-aware models requires a bounding box for each word. PDF.js cannot reliably provide this data because of its phrase-level bounding boxes instead of word-level and empty bounding boxes. PyMuPDF solves this issue, but the text parsed by PDF.js and PyMuPDF can differ. DocMaster uses PDF.js for frontend rendering and PyMuPDF in the backend and provides a robust method for mapping a PDF.js selection to the PyMuPDF context.

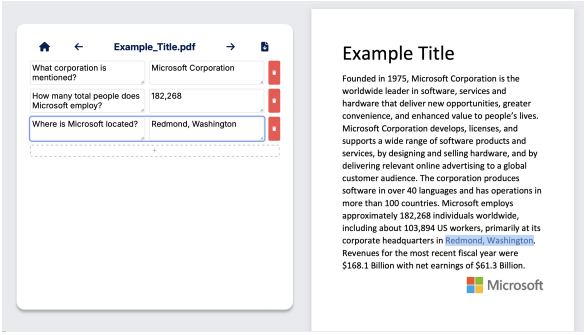



Figure 3: The annotation interface of DocMaster. The users upload a PDF/a zip of PDFs, input their questions and highlight the answers in each PDF.

servers. As such, it is configured to automatically set up and run multiple Docker containers, enabling portability across environments.

### 3.1 Annotation

The annotation interface streamlines the user process of uploading PDFs, and inputting questions and their corresponding answers, achieved through highlighting relevant text spans within the PDF. For layout models, it is essential to capture the layout information of the highlighted span. Consequently, the annotation interface must fulfill three essential requirements: (1.) accurately display the PDF, (2.) enable text highlighting, and (3.) collect layout information of the highlighted span.

We utilize Mozilla PDF.js[5] to embed the input document as a canvas onto the webpage, providing an engaging frontend experience. PDF.js incorporates an invisible *textlayer*, enabling selectable text on the canvas, enhancing the user interface. Despite its advantages, the textlayer's bounding box information, which offers layout details, presents several challenges. Firstly, it provides bounding box information primarily for spans determined by PDF.js, often encompassing entire lines and phrases but not consistently individual words. For example, in Figure 2, the user selects "*services and hardware*" and PDF.js provides bounding box information for "*leader in software, services*" and "*and hardware that deliver new*" separately, making it challenging to obtain the bounding box information for the selected text. Secondly, it occasionally detects empty spans and provides irrelevant bounding box information as shown in Figure 2. Finally, the accuracy of highlighted text detected through PDF.js is not always reliable and is susceptible to whitespace errors, as illustrated by the user selection of "*services and hardware*" in Figure 2, where spaces in the middle were not accurately preserved.

To address these limitations, we employ PyMuPDF[6] on the backend, a Python library that consistently provides word-level bounding boxes with accuracy. While PyMuPDF excels in providing accurate bounding box information, it cannot render PDFs on the webpage, hindering user-friendly text highlighting. Consequently, we integrate PDF.js in the frontend and PyMuPDF in the backend, leveraging the strengths of both. However, this integration introduces a compatibility challenge, requiring the conversion of user selec-

[5] https://mozilla.github.io/pdf.js/

[6] https://pymupdf.readthedocs.io/en/latest/

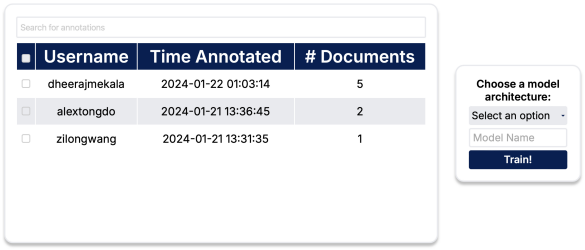



Figure 4: In training interface, the users can select one of the base models and train it using the previously annotated documents. Each row in the table indicates an annotation session and shows the number of documents annotated during that session.

tions from PDF.js context to the corresponding selections in the PyMuPDF context.

If the user-selected text is uniquely identifiable within the PDF.js context, locating its position in the PyMuPDF context is straightforward. However, when dealing with non-unique selections, we encounter the challenge of distinguishing among multiple substrings in the PyMuPDF context that could potentially represent the desired selection. To address this, we leverage the bounding box information provided by PDF.js, which often corresponds to the sentence or phrase containing the selected text. This allows us to narrow down the search area and focus specifically on that text for accurate identification. An example annotation interface is shown in Figure 3.

### 3.2 Training

After annotating their desired PDFs, users can pick and choose which annotations they would like to include as training data. To manage the potential high influx of PDFs, DocMaster organizes documents into sessions, affording users the choice to either fully include or exclude entire sessions. A new session is generated each time a user logs in, consolidating all annotations made during the active browsing window. Should modifications be necessary for an already annotated document, reuploading a previously annotated PDF retrieves and removes its data from the prior session, enabling the updated data to be stored in a new session.

The training interface is shown in Figure 4. In the training interface, all sessions are presented, showcasing the number of annotated documents and the corresponding times of annotation. This display streamlines the data selection process, providing transparency and accessibility. This information is shared publicly on the locally hosted DocMaster platform, fostering collaborative efforts within a team. Consequently, any user can leverage annotations performed by others to train a model, promoting team-wide collaboration.

DocMaster uses the transformers library from Huggingface (Wolf et al., 2019) for in-house training and inference. Annotations and trained model weights are saved in a local SQL database, eliminating dependence on third-party services and preserving data privacy.

### 3.3 Inference

The inference interface enables users to choose their preferred trained model and submit a set of documents for predictions. As DocMaster already leverages layout information in the annotation interface, we extend this approach to enhance user experience in the inference interface. Specifically, when users upload a new set of PDFs and questions to their QA model, DocMaster not only provides the inferred text but also a copy of the input PDF with highlighted bounding boxes corresponding to the inference. This highlighting aids users in pinpointing the location of their answers and any relevant surrounding context. Additionally, users can conveniently download the highlighted PDFs for future reference.

## 4 DocMaster: Building YOUR Document QA System

How can an organization utilize DocMaster to implement a document QA system tailored to their use case? In this section, we illustrate a hypothetical scenario where the HR department of a company seeks to improve its onboarding process through the integration of a QA system.

**Privacy-preserving** Shang Data Lab, Inc. has an HR team aiming to implement a QA system for new hires to address queries related to various onboarding documents. However, due to the sensitive nature of these documents, the HR team is cautious about utilizing third-party services considering potential data leakage (Nasr et al., 2023). Their preference is to ensure internal documents never leave their servers. Recognizing the open-sourced system DocMaster for its emphasis on privacy, Shang Data Lab, Inc. finds it to be a suitable solution meeting their specific requirements.

**Ease of Deployment** Setting up DocMaster is straightforward, involving the cloning of source

code and the execution of a single command: "`docker compose up`". Leveraging Docker, a widely adopted containerization software, Shang Data Lab, Inc. can swiftly have their own DocMaster operational within a few minutes.

**Parallel Annotation** Intending to train a QA model to aid in comprehending onboarding documents, the HR team at Shang Data Lab, Inc. allocates tasks to each team member, requiring them to generate questions for a subset of documents to curate training data. Utilizing DocMaster, each team member logs in and uploads a few onboarding PDFs to the annotation interface. Within this interface, they can annotate the answers to their questions by highlighting relevant text in the PDF. Working concurrently, the HR team successfully compiles a training dataset containing multiple questions and corresponding answers relevant to each onboarding document.

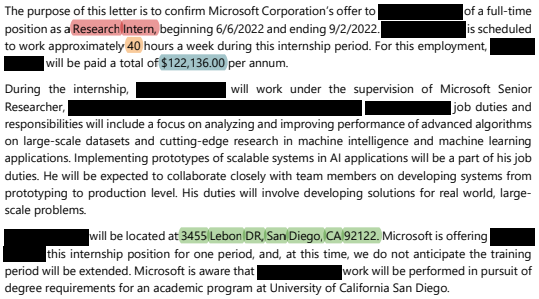
**Training & Inference** With the newly curated dataset, Shang Data Lab, Inc. initiates the training of QA models seamlessly through the training interface. Utilizing the platform's features, they have the flexibility to opt for training either a text-only or layout-aware model. Once the model is trained, they can deploy it using the inference interface, enabling new hires to leverage its capabilities. New hires can easily upload a PDF and input a set of questions, receiving not only accurate answers but also benefiting from the ability to precisely locate the answers within the document through highlighted references.

This scenario highlights the versatility of DocMaster and its aptitude to address specific needs within the AI-as-a-service ecosystem.

## 5 Public Deployment: Takeaways & Testimonials

The ISEO at UCSD[7] oversees immigration services for international students. This responsibility encompasses tasks such as certifying students' admission to full-time study programs, issuing work permits, and managing various other related processes. Each certification request undergoes a meticulous manual review of its accompanying supporting documents. During peak periods, the volume of applications can reach into the thousands.

Presently, each request is processed manually, involving a staff member who reviews supporting documents, communicates with relevant sub-teams for additional assessment, and ultimately electronically approves or declines the request. The manual nature of this process is labor-intensive and demands significant human effort. Furthermore, any delay in processing requests poses potential challenges for international students, including leaving the country or delays in commencing employment.


The purpose of this letter is to confirm Microsoft Corporation's offer to [redacted] of a full-time position as a Research Intern, beginning 6/6/2022 and ending 9/2/2022. [redacted] is scheduled to work approximately 40 hours a week during this internship period. For this employment, [redacted] will be paid a total of $122,136.00 per annum.

During the internship, [redacted] will work under the supervision of Microsoft Senior Researcher, [redacted] job duties and responsibilities will include a focus on analyzing and improving performance of advanced algorithms on large-scale datasets and cutting-edge research in machine intelligence and machine learning applications. Implementing prototypes of scalable systems in AI applications will be a part of his job duties. He will be expected to collaborate closely with team members on developing systems from prototyping to production level. His duties will involve developing solutions for real world, large-scale problems.

[redacted] will be located at 3455 Lebon DR, San Diego, CA 92122. Microsoft is offering [redacted] this internship position for one period, and, at this time, we do not anticipate the training period will be extended. Microsoft is aware that [redacted] work will be performed in pursuit of degree requirements for an academic program at University of California San Diego.

Figure 5: Highlighted answers for questions asked by ISEO office staff on a supporting document. The questions are: "*What is the job title?*" (red), "*What are the work hours per week?*" (orange), "*What is the salary or hourly rate?*" (blue), "*Where is the internship address?*" (green). Private information is redacted.

To tackle this issue, we deployed DocMaster to streamline the review process, focusing on a specific scenario: the issuance of work permits for internships, as a prototype use case. Traditionally, ISEO staff manually verifies essential fields and grants approval for work permits upon the submission of supporting documents by students. The eight key fields subject to review include employer details, salary information, job description, supervisor name & email address, weekly work hours, internship location, and start & end dates.

We formulate this as a QA task and train a model to extract necessary fields (Mekala et al., 2022b). To generate training data, four individuals without a machine learning background utilize DocMaster's annotation interface to annotate five documents each. Each annotator formulates a question for every required field and highlights the relevant answer span within the PDF (Mekala et al., 2023). The collected annotations, encompassing both text and layout information, are aggregated. Subsequently, we train two models, RoBERTa-base and LayoutLM-base, for three epochs using this annotated dataset. During the inference phase, new student documents are uploaded to the interface, and the staff member inputs questions corresponding to the required fields. The user then selects the trained model, and answers for each field are high-

[7] https://ispo.ucsd.edu/

Table 1: Performance Results on 128 applications test set in %.

| Model | Acc | F1 | Corr | Dist |
|---|---|---|---|---|
| RoBERTa-base | 76.23 | 83.77 | 93.56 | 1.13 |
| LayoutLM-base | 75.98 | 83.07 | 93.36 | 1.86 |

lighted within the PDF, as illustrated in Figure 5.

Our test set comprises 128 applications, encompassing a total of 1024 questions. After consulting with the ISEO staff, we learned that traditional QA task metrics such as exact match accuracy (Acc), f1-score (F1) alone are not sufficient; the most crucial metric for them is the average processing time of a document. The more easily identifiable the fields are, the quicker the document processing time becomes. Consequently, we tailored our automated metrics to account for this priority.

We define our correctness (Corr) metric as follows to consider partial overlaps with the ground truth. More precisely, we calculate the length of the longest contiguous matching subsequence and define a prediction as correct when the overlapping subsequence length exceeds 20% of the prediction's total length. In cases where there is no overlap, we utilize Python's difflib SequenceMatcher[8] to compute the longest contiguous matching subsequence between $P$ and $T$, excluding any "junk". A prediction is considered correct if the computed score is greater than 0.5; otherwise, it is deemed incorrect. Mathematically,

$$\text{Corr}(P,T) = \begin{cases} 1 & \text{if } \frac{\text{len}(P_{\text{indexes}} \cap T_{\text{indexes}})}{\max(\text{len}(P),\ \text{len}(T))} > 0.2 \\ 1 & \text{else if SequenceMatcher}(P,T) > 0.5 \\ 0 & \text{else} \end{cases}$$

We additionally incorporate the Euclidean distance between the predicted bounding box and the ground truth bounding box (Dist) as a performance metric. Recognizing the challenge posed by raw distance interpretation, we opt for a relative distance measurement, specifically, the distance normalized by the diagonal length of the page. A shorter distance indicates an easier identification of ground truth, leading to reduced processing time.

The performance results for RoBERTa-base and LayoutLM-base are detailed in Table 1. Notably, both models exhibit a similar performance on the test set, achieving a correctness score of approximately 94%. The disparity between exact match accuracy and correctness scores underscores the inadequacy of standard academic evaluation metrics, prompting the need for a reevaluation of metrics tailored to real-life deployment scenarios. Furthermore, we compute average bounding box distance for incorrect predictions alone, revealing values of 19.57% for RoBERTa-base and 24.39% for LayoutLM-base. This implies that when predictions are inaccurate, they tend to be in close proximity, typically within 20% of the page size, indicating correct localization despite incorrect answers.

We also measure throughput on the test set by deploying DocMaster on an AMD EPYC 7453 28-Core Processor (56 CPUs, base frequency of 2.75 GHz, boost frequency of up to 3.45 GHz). We prioritize the lightweight nature of the RoBERTa-base model over LayoutLM-base and consider it for practical deployment. Leveraging DocMaster, the ISEO experienced a sevenfold enhancement in the number of supporting documents that can be reviewed per hour, escalating from 15 to 100.

Considering the sensitivity of the information contained in supporting documents, encompassing details like home and work addresses, salary information, and supervisor details, DocMaster stands out as a fitting solution, guaranteeing the privacy of confidential data with on-device computing. Offering both high performance and convenience, with the ability to annotate data, train models, and make predictions all within a unified platform, DocMaster emerges as the optimal open-sourced platform for such use cases.

## 6 Conclusion

This work introduces DocMaster, a unified Document-QA platform designed for annotation, training, and inference while prioritizing privacy preservation. DocMaster aims to empower users to train and deploy their models for document QA purposes. Despite the availability of various models, there is a scarcity of open-sourced annotation platforms. Addressing this gap, DocMaster is presented as an open-source solution where users can annotate PDFs effortlessly by simply highlighting relevant text. The platform has demonstrated its efficacy in constructing UCSD ISEO's AI assistant, contributing to a noteworthy seven-fold reduction in document processing time. The open-sourcing of DocMaster is intended to empower businesses that necessitate in-house document QA platforms.

[8] https://docs.python.org/3/library/difflib.html

## 7 Ethical Considerations

We introduce a privacy-preserving document-QA platform and identify no ethical concerns associated with its use.

## 8 Acknowledgments

The authors thank Bryant Tan, Gilen Wu-hou, Jinya Jiang, Khai Luu for their valuable contributions. We also thank Pauline DeGuzman and Emily Stewart for their support. Finally, we thank Vaidehi Gupta and Mai ElSherief for their guidance.

## References


Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Tom Bryan, Jacob Carlson, Abhishek Arora, and Melissa Dell. 2023. Efficientocr: An extensible, open-source package for efficiently digitizing world knowledge. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 579–596.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*.

Łukasz Garncarek, Rafał Powalski, Tomasz Stanisławek, Bartosz Topolski, Piotr Halama, Michał Turski, and Filip Graliński. 2021. Lambert: Layout-aware language modeling for information extraction. In *International Conference on Document Analysis and Recognition*, pages 532–547. Springer.

Jiuxiang Gu, Jason Kuen, Vlad I Morariu, Handong Zhao, Rajiv Jain, Nikolaos Barmpalios, Ani Nenkova, and Tong Sun. 2021. Unidoc: Unified pretraining framework for document understanding. *Advances in Neural Information Processing Systems*, 34:39–50.

Teakgyu Hong, Donghyun Kim, Mingi Ji, Wonseok Hwang, Daehyun Nam, and Sungrae Park. 2022. Bros: A pre-trained language model focusing on text and layout for better key information extraction from documents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10767–10775.

Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. 2022. Layoutlmv3: Pre-training for document ai with unified text and image masking. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 4083–4091.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Kyle Lo, Zejiang Shen, Benjamin Newman, Joseph Z Chang, Russell Authur, Erin Bransom, Stefan Candra, Yoganand Chandrasekhar, Regan Huff, Bailey Kuehl, et al. 2023. Papermage: A unified toolkit for processing, representing, and manipulating visually-rich scientific documents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 495–507.

Tengchao Lv, Yupan Huang, Jingye Chen, Lei Cui, Shuming Ma, Yaoyao Chang, Shaohan Huang, Wenhui Wang, Li Dong, Weiyao Luo, et al. 2023. Kosmos-2.5: A multimodal literate model. *arXiv preprint arXiv:2309.11419*.

Dheeraj Mekala, Chengyu Dong, and Jingbo Shang. 2022a. LOPS: Learning order inspired pseudo-label selection for weakly supervised text classification. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4894–4908, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Dheeraj Mekala, Tu Vu, Timo Schick, and Jingbo Shang. 2022b. Leveraging QA datasets to improve generative data augmentation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9737–9750, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Dheeraj Mekala, Jason Wolfe, and Subhro Roy. 2023. ZEROTOP: Zero-shot task-oriented semantic parsing using large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5792–5799, Singapore. Association for Computational Linguistics.

Milad Nasr, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A Feder Cooper, Daphne Ippolito, Christopher A Choquette-Choo, Eric Wallace, Florian Tramèr, and Katherine Lee. 2023. Scalable extraction of training data from (production) language models. *arXiv preprint arXiv:2311.17035*.

Vincent Perot, Kai Kang, Florian Luisier, Guolong Su, Xiaoyu Sun, Ramya Sree Boppana, Zilong Wang, Jiaqi Mu, Hao Zhang, and Nan Hua. 2023. Lmdx: Language model-based document information extraction and localization. *arXiv preprint arXiv:2309.10952*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

QiuXing Michelle Tan, Qi Cao, Chee Kiat Seow, and Peter Chunyu Yau. 2023. Information extraction system for invoices and receipts. In *International Conference on Intelligent Computing*, pages 77–89. Springer.

Zineng Tang, Ziyi Yang, Guoxin Wang, Yuwei Fang, Yang Liu, Chenguang Zhu, Michael Zeng, Cha Zhang, and Mohit Bansal. 2023. Unifying vision, text, and layout for universal document processing.

In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19254–19264.

Zilong Wang, Jiuxiang Gu, Chris Tensmeyer, Nikolaos Barmpalios, Ani Nenkova, Tong Sun, Jingbo Shang, and Vlad Morariu. 2022. Mgdoc: Pre-training with multi-granular hierarchy for document image understanding. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3984–3993.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Yang Xu, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, Wanxiang Che, et al. 2021. Layoutlmv2: Multi-modal pre-training for visually-rich document understanding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2579–2591.

Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. 2020. Layoutlm: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1192–1200.