LayoutReader: Pre-training of Text and Layout for Reading Order Detection

Zilong Wang[1]*, **Yiheng Xu**[2]*, **Lei Cui**[2], **Jingbo Shang**[1], **Furu Wei**[2]
[1]University of California, San Diego
[2]Microsoft Research Asia
{zlwang,jshang}@ucsd.edu
{t-yihengxu,lecu,fuwei}@microsoft.com

Abstract

Reading order detection is the cornerstone to understanding visually-rich documents (e.g., receipts and forms). Unfortunately, no existing work took advantage of advanced deep learning models because it is too laborious to annotate a large enough dataset. We observe that the reading order of WORD documents is embedded in their XML metadata; meanwhile, it is easy to convert WORD documents to PDFs or images. Therefore, in an automated manner, we construct **ReadingBank**, a benchmark dataset that contains reading order, text, and layout information for 500,000 document images covering a wide spectrum of document types. This first-ever large-scale dataset unleashes the power of deep neural networks for reading order detection. Specifically, our proposed **LayoutReader** captures the text and layout information for reading order prediction using the seq2seq model. It performs almost perfectly in reading order detection and significantly improves both open-source and commercial OCR engines in ordering text lines in their results in our experiments. The dataset and models are publicly available at https://aka.ms/layoutreader.

1 Introduction

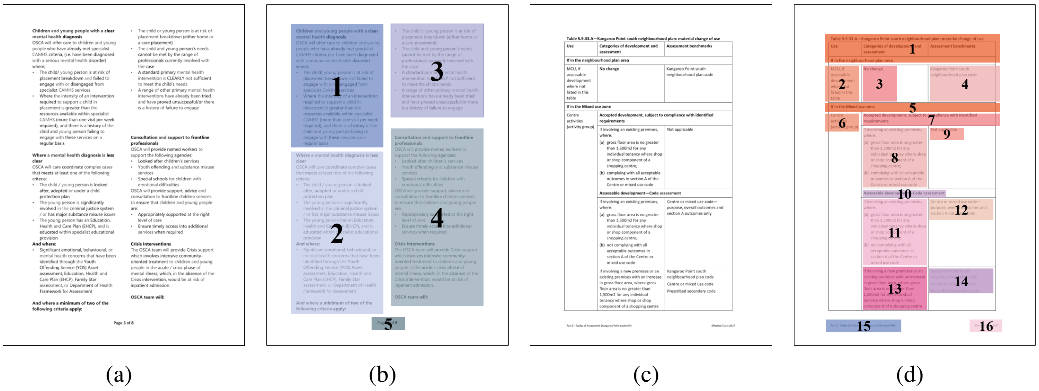
Reading order detection, aiming to capture the word sequence which can be naturally comprehended by human readers, is a fundamental task for visually-rich document understanding. Current off-the-shelf methods usually directly borrow the results from the Optical Character Recognition (OCR) engines (Xu et al., 2020) while most OCR engines arrange the recognized tokens or text lines in a top-to-bottom and left-to-right way (Clausner et al., 2013). Apparently, as shown in Figure 1, this heuristic method is not optimal for certain document types, such as multi-column templates, forms, invoices, and many others. An incorrect reading order will lead to unacceptable results for document understanding tasks such as the information extraction from receipts/invoices. Therefore, an accurate reading order detection model is indispensable to the document understanding tasks.

In the past decades, some conventional machine learning based or rule based methods (Aiello et al., 2003; Ceci et al., 2007; Malerba and Ceci, 2007; Malerba et al., 2008; Ferilli et al., 2014) have been proposed. However, these approaches are usually trained with only a small number of samples within a restricted domain or resort to unsupervised methods with empirical rules, because it is too laborious to annotate a large enough dataset. These models can barely show case studies of certain reading order scenarios and cannot be easily adapted for real-world reading order problems. Recently, deep learning models (Li et al., 2020a) have been applied to address the reading order issues for images from E-commerce platforms. Although good performance has been achieved, it is time-consuming and labor-intensive to produce an in-house dataset, while they are still not publicly available to compare with other deep learning approaches. Therefore, to facilitate the long-term research of reading order detection, it is inevitable to leverage automated approaches to create a real-world dataset in general domains, not only with high quality but also of larger magnitude than the existing datasets.

To this end, we propose ReadingBank, a benchmark dataset with 500,000 real-world document images for reading order detection. Distinct from the conventional human-labeled data, the proposed method obtains high-quality reading order annotations in a simple but effective way with automated metadata extraction. Inspired by existing document layout annotations (Siegel et al., 2018; Zhong et al., 2019; Li et al., 2020b,c), there are a large number of Microsoft WORD documents with a wide variety of templates that are available on the internet. Typically, the WORD documents have two formats: the binary format (Doc files) and the

*Contributions during internship at MSRA.

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 4735–4744
November 7–11, 2021. ©2021 Association for Computational Linguistics



(a) (b) (c) (d)

Figure 1: Document image examples in ReadingBank with the reading order information. The colored areas show the paragraph-level reading order.

XML format (DocX files). In this work, we exclusively use WORD documents with the XML format as the reading order information is embedded in the XML metadata. Furthermore, we convert the WORD documents into the PDF format so that the 2D bounding box of each word can be easily extracted using any off-the-shelf PDF parser. Finally, we apply a carefully designed coloring scheme to align the text in the XML metadata with the bounding boxes in PDFs.

With the large-scale dataset, it is possible to take advantage of deep neural networks to solve reading order detection task. We further propose LayoutReader, a novel reading order detection model in which the seq2seq model is used by encoding the text and layout information and generating the index sequence in the reading order. Ablation studies on the input modalities show that both text and layout information are essential to the final performance. The LayoutReader with both modalities surpasses other comparative methods and performs almost perfectly in reading order detection. In addition, we also adapt the results of LayoutReader to open-source and commercial OCR engines in ordering text lines. Experiments show that the line ordering of both open-source and commercial OCR engines can be greatly improved. We believe that ReadingBank and LayoutReader will empower more deep learning models in the reading order detection task and foster more customized neural architectures to push the new SOTA on this task.

The contributions are summarized as follows:

- We present ReadingBank, a benchmark dataset with 500,000 document images for reading order detection. To the best of our knowledge, this is the first large-scale benchmark for the research of reading order detection.
- We propose LayoutReader for reading order detection and conduct experiments with different parameter settings. The results confirm the effectiveness of LayoutReader in detecting reading order of documents and improving line ordering of OCR engines.
- The ReadingBank dataset and LayoutReader models will be publicly available to support more deep learning models on reading order detection.

Reproducibility. The code and datasets are publicly available at https://aka.ms/layoutreader.

2 Problem Formulation

Reading order refers to a well-organized readable word sequence. Although it seems a fundamental requirement of NLP datasets, it is non-trivial to obtain proper reading orders from document images due to various formats, e.g., tables, multiple columns, and most OCR engines fail to provide the proper reading order.

To solve this problem, we address the reading order detection task, aiming to extract the natural reading sequence from document images. Specifically, given a visually-rich document image $\mathcal{D}$, we acquire discrete token set $\{t_1, t_2, t_3, ...\}$ where each token t_i consists of a word w_i and the its bounding box coordinates $(x_0^i, y_0^i, x_1^i, y_1^i)$ (the left-top corner and right-bottom corner). Equipped with the textual and layout information of the tokens in the document image, we intend to sort the tokens into the reading order.

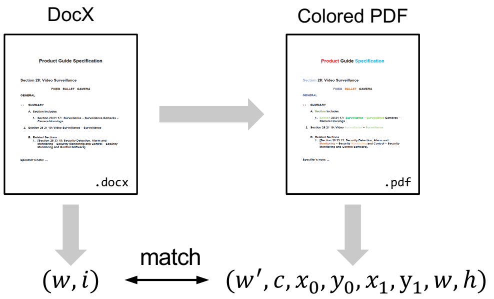


Figure 2: Building pipeline of ReadingBank, where (w, i) is the pair of word and its appearance index and $(w', c, x_0, y_0, x_1, y_1, w, h)$ is the word, word color and layout information.

3 ReadingBank

ReadingBank includes two parts, the word sequence and its corresponding bounding box coordinates. We denote the word sequence as Reading Sequence that is extracted from DocX files. The corresponding bounding boxes are extracted from the PDF files which are generated from DocX files. We propose a coloring scheme to solve the word duplication when we match each word and its bounding box.

In this section, we introduce the data pipeline in detail, including document collection, reading sequence extraction, and layout alignment with the coloring scheme. The current ReadingBank totally includes 500,000 document pages, where the training set includes 400,000 document pages and both the validation set and the test set include 50,000 document pages, respectively.

3.1 Document Collection

We crawl the WORD documents in DocX format from the internet considering the robots exclusion standard as well as the public domain license. [1] We further use the language detection API [2] with a high confidence threshold to filter non-English or bilingual documents because we focus on the reading order detection for English documents in this work. The reading order detection of other languages will be our future work. We only keep the pages with more than 50 words to guarantee the enough information on each page. In this way, we have totally collected 210,000 WORD documents in English and each page in the documents is informative enough. We further randomly select 500,000 pages to build our dataset.

3.2 Reading Sequence Extraction

The reading order in ReadingBank refers to the order of words in the DocX files. Each DocX file is a compressed archive where its word sequence can be parsed from its internal Office XML code. We adopt an open source tool `python-docx`[3] to parse the DocX file and extract the word sequence from the XML metadata. The tool also enables us to change the words' color for the layout alignment step.

We first extract the paragraphs and the tables sequentially from the parsing result. Then we traverse the paragraphs line by line and the tables cell by cell and obtain the word sequence in the DocX file. We denote the sequence as $[w_1, w_2, ..., w_n]$, where n is the number of words in this document. The obtained sequence is the reading order without the layout information and is denoted as the Reading Sequence. We would align the bounding box to each word in this sequence in the following steps.

3.3 Layout Alignment with Coloring Scheme

In our extensive collection, the same word may appear multiple times in the same document, and we need to solve this duplication when we assign the coordinates to each word. Therefore, we give each word an extra label indicating its appearance index. For example, given a sequence [the, car, hits, the, bus], the extra labels should be [0, 0, 0, 1, 0] since there are two "the"s in this example. In this way, each pair of the word and its appearance index is unique and can serve as the key when assigning the location coordinates.

Meanwhile, we propose the coloring scheme to show the keys in the DocX file without changing the original layout pattern. We map the appearance index to the RGB colors through $\mathcal{C} : \mathbb{N} \mapsto \mathbf{RGB}$ and color the words accordingly. To eliminate the interference from the original word color, we first color all the words into black.

$$
\begin{aligned}
r &= i\&0\text{x}110000 \\
g &= i\&0\text{x}001100 \\
b &= i\&0\text{x}000011 \\
\mathcal{C}(i) &= (\mathbf{R}: r, \mathbf{G}: g, \mathbf{B}: b)
\end{aligned}
$$

[1] More ethical details are included in the Ethical Consideration section.

[2] https://azure.microsoft.com/en-us/services/cognitive-services/text-analytics/

[3] https://pypi.org/project/python-docx/

Split	#Word Avg.	Avg. BLEU	ARD	BLEU Distribution			
				(0.00, 0.25]	(0.25, 0.50]	(0.50, 0.75]	(0.75, 1.00]
Train	196.38	0.6974	8.4708	9,666 2.42%	58,785 14.70%	155,662 38.92%	175,884 43.97%
Validation	196.02	0.6974	8.5140	1,203 2.41%	7,351 14.70%	19,387 38.78%	22,053 44.11%
Test	196.55	0.6972	8.4569	1,232 2.46%	7,329 14.66%	19,555 39.10%	21,893 43.78%
All	196.36	0.6974	8.4737	12,101 2.42%	73,465 14.69%	194,604 38.92%	219,830 43.97%

Table 1: Dataset statistics of training, validation, and test sets in ReadingBank. The BLEU and ARD scores are calculated for the left-to-right and top-to-bottom order to measure the difficulty of training samples

where i is the appearance index of the given word; & is the bit-wise and operation; $\mathcal{C}$ is the mapping function.

Although DocX files provide a reasonable reading sequence but the location of each word in DocX files is not fixed. Therefore, we use the PDF files produced by the colored DocX files as an intermediate to extract layout information. We adopt `PDF Metamorphosis .Net`[4] to convert the DocX files to PDF and use an open source tool `MuPDF`[5] as the PDF parser. We extract the words, bounding box coordinates, word color from the PDF file. Since the mapping function $\mathcal{C}$ is a one-to-one correspondence, we easily get the appearance index by using the coloring scheme. For the convenience of future study, we also extract the height and width of the page. In this way, we can build a one-to-one matching between the Reading Sequence and the PDF layout information.

$$(w, i) \leftrightarrow (w', c, x_0, y_0, x_1, y_1, W, H)$$
$$\text{subject to } w = w'; c = \mathcal{C}(i)$$

where w and w' are the word in DocX and PDF, respectively; i is the appearance index of w; c is the word color recognized by PDF parser; x_0, y_0, x_1, y_1 are the left-top and right-bottom coordinates; W, H are the width and height of the page where the word locates. In the post-processing stage, we collect data for each page and build our dataset.

3.4 Dataset Statistics

The ReadingBank consists of 500,000 document pages including the image and the sequence of words and coordinates in reading order. We divide the whole dataset by ratio 8:1:1 for training, validation, and testing. Table 1 shows the details of the three subsets. The average word number, the average sentence-level BLEU score, the average relative distance score (ARD) and the sentence-level BLEU score distribution are reported. The average relative distance score (ARD) calculates the relative distance between the common elements between two sequences[6]. The BLEU and ARD scores are calculated for the left-to-right and top-to-bottom order using the groundtruth reading order as the reference, so as to measure the difficulty of training samples. To guarantee the data balance, the distribution of word number and BLEU score are consistent as we randomly gather pages into each subset. We assume the ReadingBank will not suffer from the data unbalance during pre-training or fine-tuning.

Since the ReadingBank is generated in an automated manner, we further conduct human evaluation to study the dataset quality. We sample 20 pages from the ReadingBank and compare them with the human annotations. The average page-level BLEU score is 0.9839 and the ARD score is 0.4473 [6], which indicates that the ReadingBank is highly consistent with the human annotations.

4 LayoutReader

With the ReadingBank, we further propose LayoutReader to solve the reading order detection task. LayoutReader is a sequence-to-sequence model using both textual and layout information, where we leverage the layout-aware language model LayoutLM (Xu et al., 2020) as encoder and modify the generation step in the encoder-decoder structure to

[4] `https://sautinsoft.com/products/pdf-metamorphosis/`

[5] `https://www.mupdf.com/`

[6] For details about the BLEU and ARD scores, please refer to Section 5.3

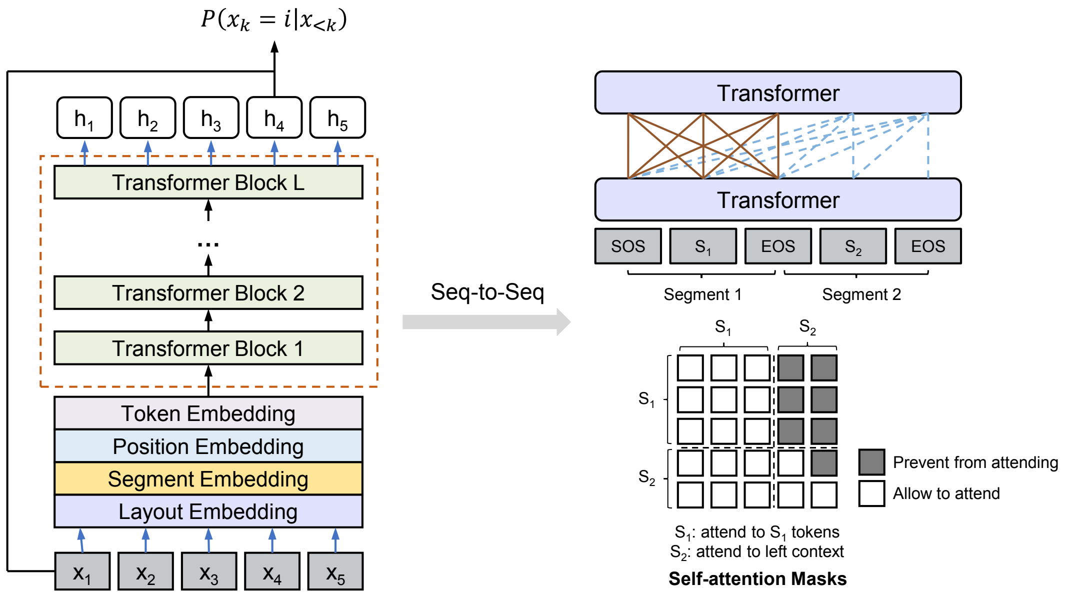


Figure 3: LayoutReader architecture for the reading order detection. The self-attention is designed for sequence-to-sequence modeling and the generation step is modified to predict the indices in the source segment.

generate the reading order sequence.

LayoutLM is a layout-aware pre-trained language model for tasks in document pages with both text and bounding boxes from OCR. It first normalizes and rounds the bounding box coordinates into integers from 0 to 1000. Then coordinates are embedded as trainable vectors like word embeddings. This new embedding layer is then added to BERT (Devlin et al., 2018). LayoutLM is first initialized with BERT and then further pre-trained with masked language model task and document classification.

Encoder: In the encoding stage, LayoutReader packs the pair of source and target segments into a contiguous input sequence of LayoutLM and carefully designs the self-attention mask to control the visibility between tokens. As shown in Figure 3, LayoutReader allows the tokens in the source segment to attend to each other while preventing the tokens in the target segment from attending to the rightward context. If 1 means allowing and 0 means preventing, the detail of the mask M is as follows:

$$M_{i,j} = \begin{cases} 1, & \text{if } i < j \text{ or } i, j \in \text{src} \\ 0, & \text{otherwise} \end{cases}$$

where i, j are the indices in the packed input sequence, so they may be from source or target segments; $i, j \in$ src means both tokens are from source segment.

Decoder: In the decoding stage, since the source and target are reordered sequences, the prediction candidates can be constrained to the source segment. Therefore, we ask the model to predict the indices in the source sequence. The probability is calculated as follows:

$$\mathcal{P}(x_k = i|x_{<k}) = \frac{\exp{(e_i^T h_k + b_k)}}{\sum_j \exp{(e_j^T h_k + b_k)}}$$

where i is an index in the source segment; e_i and e_j are the i-th and j-th input embeddings of the source segment; h_k is the hidden states at the k-th time step; b_k is the bias at the k-th time step.

5 Experiments

We introduce the comparative methods, implementation details, and evaluation metrics for the experiments. We design three experiments for LayoutReader on ReadingBank, including reading order detection, input order study, and adaption on OCR engines. In addition, we also show the real-world examples in the case study.

5.1 Comparative Methods

LayoutReader considers both text and layout information with the multi-modal encoder LayoutLM. To further study the role of each modality, we design two comparative models, including LayoutReader (layout only) and LayoutReader (text only). We also report the results of the Heuristic Method as our baseline.

Method	Encoder	Avg. Page-level BLEU ↑	ARD ↓
Heuristic Method	-	0.6972	8.46
LayoutReader (text only)	BERT	0.8510	12.08
	UniLM	0.8765	10.65
LayoutReader (layout only)	LayoutLM (layout only)	0.9732	2.31
LayoutReader	LayoutLM	**0.9819**	**1.75**

Table 2: Evaluation results of the LayoutReader on the reading order detection task, where the source-side of training/testing data is in the left-to-right and top-to-bottom order

Heuristic Method: This method refers to sorting words from left to right and from top to bottom.

LayoutReader (text only): We replace LayoutLM with textual language models, e.g. BERT (Devlin et al., 2018), UniLM (Dong et al., 2019), which means LayoutReader (text only) predicts the reading order only through textual information. Our experiments build two versions of LayoutReader (text only), which use BERT or UniLM as a substitute of LayoutLM, respectively.

LayoutReader (layout only): We remove the token embeddings in LayoutLM. The token embeddings are vital for Transformer to extract textual information. After removing these embeddings, LayoutReader (layout only) only considers the 1D and 2D positional layout information.

5.2 Implementation Details

Our implementation is built upon the HuggingFace Transformers (Wolf et al., 2019) and the LayoutReader is implemented with the s2s-ft toolkit from the repository of Dong et al. (2019)[7]. The pre-trained models used are in their base version. We use 4 Tesla V100 GPUs with batch size of 4 per GPU during training. The number of training epochs is 3 and the training process takes approximately 6 hours. We optimize the models with the AdamW optimizer. The initial learning rate is 7×10^{-5} and the number of warm-up steps is 500.

5.3 Evaluation Metrics

Average Page-level BLEU: The BLEU score (Papineni et al., 2002) is widely used in sequence generation. Since LayoutReader is built on a sequence-to-sequence model, it is natural to evaluate our models with BLEU scores. BLEU scores measure the n-gram overlaps between the hypothesis and reference. We report Average Page-level BLEU in our experiments. The page-level BLEU refers to the micro-average precision of n-gram overlaps within a page.

Average Relative Distance (ARD): The ARD score is proposed to evaluate the difference between reordered sequences. It measures the relative distance between the common elements in the different sequence. Since our reordered sequence is generated, the ARD allows the element omission but adds a punishment for it. Given a sequence $A = [e_1, e_2, ..., e_n]$ and its generated reordered sequence $B = [e_{i_1}, e_{i_2}, ..., e_{i_m}]$, where $\{i_1, i_2, ..., i_m\} \subseteq \{1, 2, ..., n\}$, the ARD score is calculated as follows:

$$s(e_k, B) = \begin{cases} |k - I(e_k, B)|, & \text{if } e_k \in B \\ n, & \text{otherwise} \end{cases}$$

$$\text{ARD}(A, B) = \frac{1}{n} \sum_{e_k \in A} s(e_k, B)$$

where e_k is the k-th element in sequence A; $I(e_k, B)$ is the index of e_k in sequence B; n is the length of sequence A.

5.4 Reading Order Detection

We train the models with left-to-right and top-to-bottom ordered inputs and report the evaluation results on the test set of ReadingBank in Table 2. We also report the results of the heuristic method. The results show that LayoutReader is superior and achieves the SOTA results compared with other baselines. It improves the average page-level BLEU by 0.2847 and decreases the ARD by 6.71. Even if we remove some of the input modalities, there is still 0.16 and 0.27 improvements of BLEU in LayoutReader (text only) and LayoutReader (layout only), and there is a steady 6.15 reduction of ARD in LayoutReader (layout only). However, we also see a drop of ARD in LayoutReader (text only), mainly because of the severe punishment in ARD for token omission (see ARD

[7] https://github.com/microsoft/unilm/tree/master/s2s-ft

Method	Avg. Page-level BLEU ↑			ARD ↓		
	r=100%	r=50%	r=0%	r=100%	r=50%	r=0%
LayoutReader (text only, BERT)	0.3355	0.8397	0.8510	77.97	15.62	12.08
LayoutReader (text only, UniLM)	0.3440	0.8588	0.8765	78.67	13.65	10.65
LayoutReader (layout only)	0.9701	0.9729	0.9732	2.85	2.61	2.31
LayoutReader	**0.9765**	**0.9788**	**0.9819**	**2.50**	**2.24**	**1.75**

Table 3: Input order study with left-to-right and top-to-bottom inputs in evaluation, where r is the proportion of shuffled samples in training.

Method	Avg. Page-level BLEU ↑			ARD ↓		
	r=100%	r=50%	r=0%	r=100%	r=50%	r=0%
LayoutReader (text only, BERT)	0.3085	0.2730	0.1711	78.69	85.44	**67.96**
LayoutReader (text only, UniLM)	0.3119	0.2855	0.1728	80.00	85.60	71.13
LayoutReader (layout only)	0.9718	0.9714	0.1331	2.72	2.82	105.40
LayoutReader	**0.9772**	**0.9770**	**0.1783**	**2.48**	**2.46**	72.94

Table 4: Input order study with token-shuffled inputs in evaluation, where r is the proportion of shuffled samples in training.

definition). LayoutReader (text only) can guarantee the right order of tokens but suffers from the incompleteness of generation. We also conclude that the layout information plays a more important role than textual information in the reading order detection. LayoutReader (layout only) surpasses the LayoutReader (text only) greatly by about 0.1 in BLEU and about 9.0 in ARD.

5.5 Input Order for Training and Testing

We shuffle the input tokens of sequence-to-sequence model in a certain proportion of training samples to study the accuracy of LayoutReader for different input orders. The proportion of token-shuffled training samples is denoted as r. We build three versions of comparative models with r equaling 100%, 50% and 0%. The left-to-right and top-to-bottom order provide remarkable hints for reading order detection. However, in this input order study, these hints are incomplete during training. We design two evaluation methods. Table 3 shows the results when we evaluate the comparative models with left-to-right and top-to-bottom inputs. Table 4 shows the results when we evaluate the comparative models with token-shuffled inputs.

From Table 3, we observe that LayoutReader (layout only) and LayoutReader are more robust to the shuffled tokens during training, and all three comparative models perform well with the left-to-right and top-to-bottom inputs in evaluation. We attribute it to the consideration of layout information, which is consistent under shuffling.

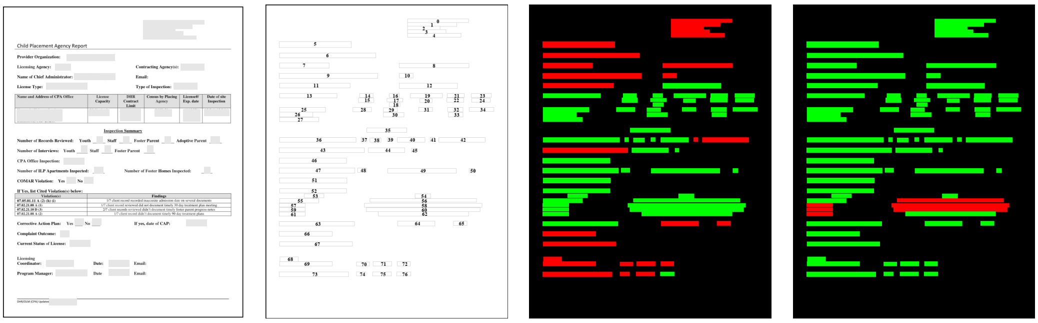
From Table 4, we see a drop when we train LayoutReader with $r = 0\%$ token-shuffled inputs and evaluate it with all token-shuffled inputs. We explain that models trained on $r = 0\%$ token-shuffled inputs tend to overfit the left-to-right and top-to-bottom order due to overlaps between this order and groundtruth, while the token-shuffled inputs in evaluation are totally unseen to these models.

5.6 Adaption to OCR Engines

Most OCR engines provide reading order information for the text lines, where some of them may be problematic. To improve the text line ordering, we extend the token-level reading order to text lines and adapt it to OCR engines.

We first assign each token in our token-level order to the text lines according to the percentage of spatial overlapping. Given a token bounding box b and a text line bounding box B, the token is assigned to the text line which overlaps the most with the token, i.e. $\hat{B} = \text{argmax}_B(B \cap b)$, where $\cap$ means spacial overlapping. Then we calculate the minimum of token indices in each text line as its ranking value and produce an improved text line order from the token-level order.

It should be noted that the token-level order can be the order given by ReadingBank or the result generated by LayoutReader. Therefore, we build a text line ordering groundtruth by adapting the ReadingBank to text lines and evaluate the performance



(a) Original image (b) Groundtruth (c) The commercial OCR (d) LayoutReader

Figure 4: Case Study: (a) is the original image (some fields are masked because of privacy); (b) is the text line reading order groundtruth from ReadingBank Adaption; (c) and (d) are the results of a commercial OCR engine and LayoutReader Adaption where green and red denote the correct and incorrect predicted indices.

Method	Avg. Page-level BLEU ↑	ARD ↓
Heuristic Method	0.3391	13.61
Tesseract OCR	0.7532	1.42
LayoutReader	**0.9360**	**0.27**

Table 5: Adaption to text lines of Tesseract OCR

Method	Avg. Page-level BLEU ↑	ARD ↓
Heuristic Method	0.3752	10.17
The commercial OCR	0.8530	2.40
LayoutReader	**0.9430**	**0.59**

Table 6: Adaption to text lines of the commercial OCR

of LayoutReader in text line ordering accordingly. We also report the performance of the Heuristic Method and OCR engines. We conduct experiments with two OCR engines, including an open source OCR engine Tesseract, and a cloud-based commercial OCR API. The results are shown in Table 5 and Table 6. We can see a great improvement with LayoutReader Adaption. This experiment further demonstrates the effectiveness and extends the application of LayoutReader.

5.7 Case Study

We select a representative example from our test set and show the text line orders in Figure 4. We compare the text line order of the commercial OCR engine and LayoutReader Adaption with the groundtruth from ReadingBank Adaption. We visualize the results with colors where green and red denotes correct and incorrect results. We see LayoutReader Adaption improves the text line ordering of the OCR engine, which is consistent with our results in Section 5.6.

6 Related Work

Reading order detection was first proposed in (Aiello et al., 2003), where they used a propositional language of qualitative rectangle relations to detect reading order from document images. This is also considered as the first rule-based reading order detection system. With the development of machine learning methods, (Ceci et al., 2007) proposed a probabilistic classifier using the Bayesian framework and reconstructing either single or multiple chains of layout components. Meanwhile, (Malerba and Ceci, 2007) applied an ILP learning algorithm to introduce the definitions of the two predicates and establish an ordering relationship. After that, (Malerba et al., 2008) investigated the problem of detecting the reading order relationship between components of a logical structure with domain specific knowledge. (Ferilli et al., 2014) presented an unsupervised strategy for identifying the correct reading order of a document page's components based on abstract argumentation. The method is based on an empirical assumption about how humans behave when reading documents. More recently, deep learning models have become the mainstream solution for many machine learning problems. (Li et al., 2020a) proposed an end-to-end OCR text reorganizing model, where they use a Graph Neural Network with an attention map to encode the text blocks with visual layout features, with an attention-based sequence decoder to reorder the OCR text into a proper sequence.

7 Conclusion

In this paper, we present ReadingBank, a benchmark dataset for reading order detection that contains 500,000 document images. In addition, we also propose LayoutReader, a novel reading order detection approach built upon the pre-trained LayoutLM model. Experiments show that the LayoutReader has significantly outperformed the left-to-right and top-to-bottom heuristics as well as several strong baselines. Furthermore, the LayoutReader can be easily adapted to any OCR engines so that the reading order can be improved for downstream tasks. The ReadingBank dataset and LayoutReader model will be publicly available to support more research on reading order detection.

For future research, we will investigate how to generate a larger synthesized dataset from the ReadingBank, where noisy information and rotation can be applied to the clean images to make the model more robust. Moreover, we will label the reading order information on a real-world dataset from scanned documents. Considering the LayoutReader model as a pre-trained reading order detection model, we will also explore whether a few human labeled samples would be sufficient for the reading order detection in a specific domain.

A Ethical Consideration

The ethical impact of our research has always been an important consideration. While pursuing better performance and high quality datasets, we respect the intellectual property of the data resources. We sincerely hope our research will benefit the academia and foster more related study and, meanwhile, all ethical standards are strictly followed.

When building the new dataset, ReadingBank, we carefully crawl the public available data from the internet. We strictly follow the robots exclusion standard of each website to make sure we are permitted to collect the data. We also exclude the web pages with privacy issues and only keep those pages we have the permission to edit and redistribute according to the license rules. To guarantee there is no potential ethical violation, we will publicize a proportion of our dataset (about 100 pages) and this subset will be manually checked and redacted while the access of the whole version requires our further permission. All the data in our dataset will be protected by Apache 2.0 license.

We design the reading order detection as a fundamental task for the document image understanding. Numerous following tasks can be built on the basis of it. We do not set preference or limitation about the areas when we crawl the data so we believe the result of LayoutReader can be well generalized to other visually-rich document images due to the vast scope our dataset covers.

References

Marco Aiello, A Smeulders, et al. 2003. *Bidimensional relations for reading order detection*. University of Groningen, Johann Bernoulli Institute for Mathematics and Computer Science.

M. Ceci, M. Berardi, G. Porcelli, and D. Malerba. 2007. A data mining approach to reading order detection. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, volume 2, pages 924–928.

C. Clausner, S. Pletschacher, and A. Antonacopoulos. 2013. The significance of reading order in document recognition and its evaluation. In *2013 12th International Conference on Document Analysis and Recognition*, pages 688–692.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 13042–13054.

Stefano Ferilli, Domenico Grieco, Domenico Redavid, and Floriana Esposito. 2014. Abstract argumentation for reading order detection. In *Proceedings of the 2014 ACM Symposium on Document Engineering*, DocEng '14, page 45–48, New York, NY, USA. Association for Computing Machinery.

Liangcheng Li, Feiyu Gao, Jiajun Bu, Yongpan Wang, Zhi Yu, and Qi Zheng. 2020a. An end-to-end ocr text re-organization sequence learning for rich-text detail image comprehension. In *Computer Vision – ECCV 2020*, pages 85–100, Cham. Springer International Publishing.

Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, Ming Zhou, and Zhoujun Li. 2020b. Tablebank: Table benchmark for image-based table detection and recognition. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 1918–1925.

Minghao Li, Yiheng Xu, Lei Cui, Shaohan Huang, Furu Wei, Zhoujun Li, and Ming Zhou. 2020c. Docbank: A benchmark dataset for document layout analysis. *arXiv preprint arXiv:2006.01038*.

Donato Malerba and Michelangelo Ceci. 2007. Learning to order: A relational approach. In *International Workshop on Mining Complex Data*, pages 209–223. Springer.

Donato Malerba, Michelangelo Ceci, and Margherita Berardi. 2008. Machine learning for reading order detection in document image understanding. In *Machine Learning in Document Analysis and Recognition*, pages 45–69. Springer.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Noah Siegel, Nicholas Lourie, Russell Power, and Waleed Ammar. 2018. Extracting scientific figures with distantly supervised neural networks. *Proceedings of the 18th ACM/IEEE on Joint Conference on Digital Libraries*.

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*.

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*, KDD '20, page 1192–1200, New York, NY, USA. Association for Computing Machinery.

Xu Zhong, Jianbin Tang, and Antonio Jimeno-Yepes. 2019. Publaynet: Largest dataset ever for document layout analysis. *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1015–1022.