Learning Reading Order via Document Layout with Layout2Pos

Anonymous ACL submission

Abstract

Due to their remarkable performance, general-purpose multimodal pre-trained language models have gained widespread adoption for Document Understanding tasks. The majority of pre-trained language models rely on serialized text, extracted using either Optical Character Recognition (OCR) or PDF parsing. However, accurately determining the reading order of visually-rich documents (VrDs) is challenging, potentially affecting the accuracy of the extracted text and leading to sub-optimal performance in downstream tasks. For information extraction tasks, where entity recognition is commonly framed as a sequence-labeling task, incorrect reading order can hinder entity labeling. In this work, we avoid reading order issues by discarding sequential position information. Based on the intuition that layout contains the information for correct reading order, we present Layout2Pos – a shallow Transformer designed to generate position embeddings from layout. Incorporated into a BART architecture, our approach demonstrates competitiveness with models dependent on reading order across three benchmark datasets for information extraction. We also show that evaluating models using a reading order different from the one seen during training can result in substantial performance drops, thereby highlighting the importance of not relying on the reading order of documents.

1 Introduction

The organization of textual content in a specific layout is crucial for conveying information, holding significant importance across various written materials, including business documents, scholarly papers, and news articles. In particular, layout determines the sequence in which text is intended to be read or processed within a document, *i.e.*, the *reading order*. A well-designed reading order ensures that readers can follow the logical flow and structure of information and comprehend the intended meaning of the text. However, defining a proper reading order is non-trivial due to the complexity of document layouts, which may include elements such as tables and multiple columns.

When language or information extraction models are trained with a reading order that aligns with human understanding, they learn to capture the relationships between words, sentences, and paragraphs. Hence, reading order is crucial for models to perform well. Most pre-training methods for Document Understanding rely on serialized text, where either an Optical Character Recognition (OCR) engine or a PDF parser is used to extract text. However, due to the variety of layout formats, most OCR engines and PDF parsers struggle to provide accurate reading orders, introducing *serialization errors*. Serialization errors, *i.e.*, noise that may arise during text extraction, such as misinterpretations or omissions, can impact the accuracy of the extracted text and, therefore, affect the entire text processing pipeline. Without an accurate reading order, models may misinterpret the relationships between different parts of the text. This poses a substantial challenge in Document Understanding, where document layouts can be complex.

Specifically, in information extraction tasks from visually-rich documents (VrDs), also referred to as *visual information extraction*, the primary goal[1] is to identify entities of predefined semantic types (e.g., names, dates, addresses). In this context, performance is notably impacted by serialization errors. Following the classic settings of NLP, the task is commonly framed as a sequence-labeling problem. This approach involves labeling each token using a tagging scheme, such as BIO-tagging (Ramshaw and Marcus, 1999), and leveraging these tags to identify entities. A sequence labeling-based

[1]Additionally, the task extends to classifying the relationships between these recognized entities (*relation extraction*). In this work, we do not focus on this task.

approach operates under the assumption that each identified segment of an entity forms a continuous sequence of words within the input. While this assumption is valid for plain texts, it may not hold for real-world documents, where OCR systems or PDF parsers might not correctly organize text (e.g., an entity might be split into non-continuous fragments). Such disordered input disrupts the BIO-tagging scheme, preventing the models from accurately identifying entities.

On the other hand, layout inherently encapsulates the correct reading order of documents by visually organizing content in a structured manner. A well-designed layout guides the reader's natural progression from one section to another, ensuring logical information flow. Therefore, understanding the layout provides essential cues for determining the correct reading order. Yet, existing pre-training methods for Document Understanding often neglect this aspect, opting to oversimplify the integration of layout by, e.g., adding it as an extra input embedding (Xu et al., 2020b).

We focus on mitigating serialization errors by *entirely discarding sequential position information*. We introduce *Layout2Pos*, a shallow Transformer designed to generate position embeddings from the document layout. Our endeavor is twofold: from a practical standpoint, we aim to enhance the robustness of models to reading order changes, crucial for real-world applications; from a theoretical perspective, we demonstrate that it is feasible to discard sequential position information without compromising overall performance. We integrate this module into a sequence-to-sequence framework. To train the model, the language modeling task is coupled with a pre-training strategy designed to instill the model with the ability to learn the reading order from layout information. This integration eliminates the reliance on reading order and enables the generation of values that are not explicitly present in the input. We demonstrate the benefits of our approach for visual information extraction tasks, showcasing competitive performance to models that depend on reading order.

2 Related Work

Multimodal Pre-trained Language Models To process VrDs in document understanding tasks, various approaches have proposed to incorporate layout information into language models, leveraging the modeling capabilities of Transformers (Vaswani et al., 2017). LayoutLM (Xu et al., 2020b) is the first to encode layout information with learned 2D position embeddings obtained from word bounding boxes. Extending the concept of relative position bias (Raffel et al., 2020) to the 2D scenario, LayoutLMv2 (Xu et al., 2020a) builds upon LayoutLM by adding bias terms to the attention scores, encoding the 2D relative position of tokens with respect to each other. DocFormer (Appalaraju et al., 2021) facilitates the correlation between text and images by sharing learned spatial embeddings across modalities. However, these methods rely on an OCR-induced reading order, which may not align with human reading patterns. Rather than treating layout information as an extra feature, we leverage it to learn position embeddings, removing the need for sequential position information obtained through OCR.

Addressing Reading Order Issues To address serialization errors, automatic word reordering techniques can be employed. ERNIE-Layout (Peng et al., 2022) uses an in-house document layout analysis toolkit that provides an appropriate reading order based on the spatial distribution of words, pictures, and tables. Enhanced with this knowledge, the token sequence can be rearranged in a way that aligns better with human reading patterns compared to the raster-scan order, which rearranges tokens from the top-left to the bottom-right corner. LayoutReader (Wang et al., 2021) is a sequence-to-sequence model, employing LayoutLM as its encoder, that generates the reading order of documents and improves the line ordering capabilities of OCR engines. XYLayoutLM (Gu et al., 2022) introduces an augmentation algorithm based on XY Cut (Ha et al., 1995) to generate a series of proper reading orders for training. Additionally, it adaptively generates position embeddings based on input lengths using dilated convolutions to extract local layouts. Token Path Prediction (Zhang et al., 2023) frames visual information extraction tasks as the prediction of token paths within a complete directed graph of tokens, using a prediction head compatible with Transformer-based language models. Our approach shares a similar module-based structure, while introducing more robust 1D position encodings learned from the spatial position of tokens in the document page.

Generative Methods for Information Extraction Unlike sequence labeling approaches that entirely depend on the content extracted via OCR, generative models can generate text without being restricted by the document's content or its reading order, enabling them to potentially correct OCR-induced errors. Sage et al. (2020) represent the information to be extracted as a sequence of tokens in the XML language. They employ a recurrent encoder-decoder architecture to generate XML representations, using pointer-generator networks (See et al., 2017) to allow the model to dynamically decide whether to generate a word from its vocabulary or copy it directly from the document. Townsend et al. (2021) use a Transformer language model trained on database records to generate JSON-like representation of the extracted information. In close relation to our work, TILT Powalski et al. (2021) is a Transformer encoder-decoder model enhanced with layout and visual information, specifically designed for information extraction tasks from VrDs. Instead of relying on OCR for text extraction, Donut (Kim et al., 2022) uses a Transformer visual encoder to extract features from a document image. A textual Transformer decoder is then used to map these features to a desired structured format, such as JSON, for visual information extraction tasks. In contrast to Donut, our method does not rely on visual features, offering better computational efficiency when various information extraction tasks are conducted or/and complex documents are processed. Our approach eliminates dependence on OCR-induced reading order by leveraging the spatial position of tokens on the page.

3 Preliminary Experiments: OCR Serialization Errors

To gauge the extent of OCR-induced serialization errors, we conduct preliminary experiments comparing the annotated ground-truth reading order against the reading orders produced by 1) Tesseract OCR (Kay, 2007), a widely-used OCR engine, and 2) DocTR (Mindee, 2021), an OCR engine based on deep learning models. The goal is to assess the alignment of the reading order produced via OCR with the actual human reading patterns.

We use a subset of 100 documents from the ReadingBank (Wang et al., 2021) dataset. ReadingBank is a benchmark dataset for reading order detection that includes high-quality reading order annotations which capture the correct sequence of words as visually presented in the documents. Upon examination of the samples and discerning patterns that appear most frequently, we have identified four prevalent document layout types: *plain* layout, *lists*, *multicolumn* layout, and *tables*. We provide examples in Section A of the Appendix.

	Layout Type			
	Plain	**Lists**	**Multi-column**	**Tables**
Tesseract OCR	78.71	72.75	61.43	36.97
DocTR	86.83	77.83	82.54	66.11
Layout2Pos	**99.44**	**97.45**	**94.29**	**88.60**

Table 1: Accuracy (in %) obtained by each OCR engine, for each document layout type. Our model, Layout2Pos, is described in Section 4.2. Best results are reported in bold.

Tesseract OCR and DocTR are employed to extract and serialize text from the documents. The reading orders produced are compared against the ground-truth for discrepancies. Specifically, we evaluate accuracy by comparing, for each word in the ground-truth sequence, the actual next word with the one predicted by each OCR engine. We compute the accuracy obtained by each system for each specific layout type. Results are reported in Table 1. Additionally, we include the results obtained by our approach, Layout2Pos. Our findings indicate that both OCR engines face increased difficulty in reconstructing the correct reading order as the document layout becomes more complex.[2] Our approach exhibits a similar trend, although to a significantly lesser degree. Furthermore, it demonstrates higher accuracy for each document type compared to both OCR engines.

4 Reconstructing Positional Information from 2D Positions

Building on the insights gained from the previous experiment, retrieving the correct reading order is a significant challenge for OCR engines. We argue that it is possible to retrieve the correct reading order by leveraging document layout. To address this, we propose *Layout2Pos*, a transformer-based module that does not rely on the reading order generated by OCR, and learns position embeddings solely from the spatial positions of tokens.

4.1 Encoding Layout Information

To encode layout information, we use 1) bounding box information, 2) 2D relative positions, and 3)

[2] This difficulty in accurately predicting the next word is further attributed to the OCR engines' misinterpretation of certain words.

a novel method based on line and column relative positions.

Encoding Bounding Box Information The spatial position of a token is represented by its bounding box in the document page image, denoted as (x_0, y_0, x_1, y_1), where (x_0, y_0) and (x_1, y_1) correspond to the coordinates of the top-left and bottom-right corners, respectively. Following LayoutLMv2, we discretize and normalize these coordinates to integers within the range of $[0, ..., 1000]$. Four embedding tables are employed to encode spatial positions: LE_x and LE_y for the coordinate axes (x and y), and LE_w and LE_h for the bounding box size (width and height). In line with LayoutLMv2, the final layout embedding $\boldsymbol{\ell} \in \mathbb{R}^d$ of a token, whose bounding box is (x_0, y_0, x_1, y_1), is defined as follows ($\|$ denotes concatenation):

$$\begin{aligned} \boldsymbol{\ell} = \mathrm{LE}_x(x_0) \,\|\, \mathrm{LE}_y(y_0) \\ \|\, \mathrm{LE}_x(x_1) \,\|\, \mathrm{LE}_y(y_1) \\ \|\, \mathrm{LE}_w(x_1 - x_0) \\ \|\, \mathrm{LE}_h(y_1 - y_0), \end{aligned} \tag{1}$$

Leveraging 2D Relative Positions LayoutLMv2 encodes spatial relative positions as bias terms added to the attention scores to explicitly capture the spatial relationship between tokens. Following LayoutLMv2, for each pair of bounding boxes $((x_0, y_0, x_1, y_1), (x'_0, y'_0, x'_1, y'_1))$, we compute the horizontal distance $x'_0 - x_0$ between the left edge of each box and the vertical distance $y'_1 - y_1$ between the bottom edge of each box. In addition, we provide additional insights into the spatial relationships of tokens by computing the horizontal distance $x'_1 - x_0$ between the right edge of one box and the left edge of the other, indicating information about the combined length. Furthermore, we calculate the horizontal distance $x'_1 - x_1$ between the right edge of each box, providing information about the length of the second token in the pair.

Incorporating Line and Column Relative Positions Understanding the relative positions within columns provides information about the sequential structure of the document, aiding in distinguishing between different parts of the document. On the other hand, the relative positions within lines is valuable for documents with multicolumn layouts, offering insights into the spatial arrangement of text across columns. Hence, for each bounding box, we identify other bounding boxes that share the same line/column. This is determined by whether the horizontal/vertical line passing through the center of the box intersects with the other bounding boxes. If there is an intersection, the boxes are considered to be on the same line/column. For each token t_i, we determine its positions $p^{(l)}(i)$ and $p^{(c)}(i)$ within its corresponding line and column, using a left-to-right order for lines and a top-to-bottom order for columns. Then, we compute the relative sequential distance δ^l_{ij} and δ^c_{ij} between elements within each line and column. If they do not belong to the same line or column, the distance is set to ∞.

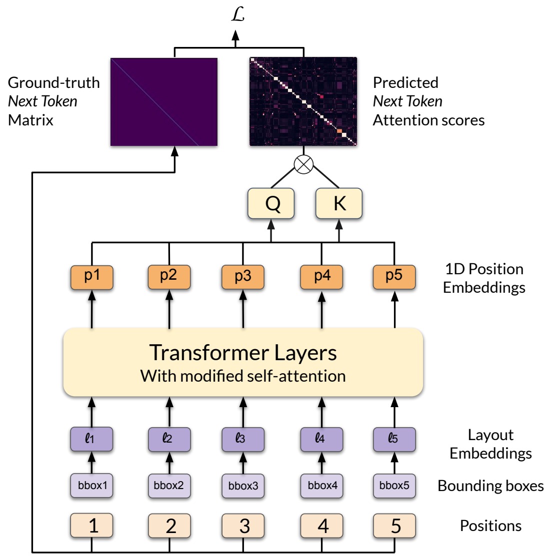


Figure 1: Layout2Pos Architecture.

Suppose $\boldsymbol{q}^\ell_i$ and $\boldsymbol{k}^\ell_i$ denote the query and key projections obtained from the layout embedding $\boldsymbol{\ell}_i$ of token i. Let $\boldsymbol{b}^{(2D_x)}$, $\boldsymbol{b}^{(2D_y)}$, $\boldsymbol{b}^{(l)}$, and $\boldsymbol{b}^{(c)}$ be the horizontal, vertical, line, and column relative position biases, respectively. In Layout2Pos, attention is re-defined as:

$$\begin{aligned} \alpha_{ij} = \frac{1}{\sqrt{d}} \left(\boldsymbol{q}^\ell_i \cdot \boldsymbol{k}^\ell_j \right) + \boldsymbol{b}^{(2D_x)}_{x^{(j)}_0 - x^{(i)}_0} + \boldsymbol{b}^{(2D_y)}_{y^{(j)}_1 - y^{(i)}_1} \\ + \boldsymbol{b}^{(2D_x)}_{x^{(j)}_1 - x^{(i)}_0} + \boldsymbol{b}^{(2D_x)}_{x^{(j)}_1 - x^{(i)}_1} + \boldsymbol{b}^{(l)}_{\delta^l_{ij}} + \boldsymbol{b}^{(c)}_{\delta^c_{ij}} \end{aligned} \tag{2}$$

4.2 Learning 1D Position Embeddings from Layout Information

Given a sequence of layout embeddings derived from token bounding box coordinates, as defined by Equation 1, Layout2Pos employs a stack of Transformer layers to contextualize the sequence. The outputs of the last layer, $\overline{\boldsymbol{\ell}}_i$, serve as position embeddings, *i.e.*, $\boldsymbol{p}_i = \overline{\boldsymbol{\ell}}_i$. The objective is for these embeddings $(\boldsymbol{p}_1, \cdots, \boldsymbol{p}_n)$ to carry information regarding the reading order. To accomplish this, we build a simple classifier on top of these embeddings, designed to compute alignment scores between each token:

$$\boldsymbol{A}_{ij} = (\boldsymbol{p}_i \boldsymbol{W}^q) \left(\boldsymbol{p}_j \boldsymbol{W}^k\right). \tag{3}$$

We assume that the attention matrix $\boldsymbol{A}$ carries information about the reading order, *i.e.*, $\boldsymbol{A}_{ij}$ represents the probability that the j-th token follows the i-th token. Let $\boldsymbol{N}$ be the ground-truth binary matrix obtained from the ground-truth reading order, where $\boldsymbol{N}_{ij}$ equals 1 if token at position j is the *next* token after token at position i in the sequence, and 0 otherwise. We define the *Next Token Position Prediction* strategy, which consists in using the attention matrix $\boldsymbol{A}$ to predict the next token of each token in the sequence (*next token matrix*). The corresponding cross-entropy loss is defined as follows:

$$\mathcal{L}_{NTPP} = -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{n} \boldsymbol{N}_{ij} \log\left(\text{softmax}_i(\boldsymbol{A}_{\cdot j})\right) \tag{4}$$

As such, Layout2Pos can be trained to capture the relationship between layout and reading order[3] by ensuring that the attention matrix $\boldsymbol{A}$ derived from the computed position embeddings carries information about the next token for each token in the sequence. The architecture of Layout2Pos is depicted in Figure 1.

4.3 Integrating Layout2Pos into a Sequence-to-Sequence Framework

Layout2Pos can be integrated into any language model, removing the reliance on sequential position information. This is achieved by substituting the traditional position encodings derived from OCR by Layout2Pos' position embeddings. Specifically, we integrate Layout2Pos into a Transformer encoder-decoder architecture, as illustrated in Figure 2. The sequence of position embeddings, obtained by Layout2Pos, is added to the sequence of token embeddings. The resulting sequence is input to the bidirectional encoder.

Corruption Loss Layout2Pos is trained together with the encoder-decoder model. While Layout2Pos learns to predict the subsequent token of each token based on layout information, the encoder-decoder follows a pre-training approach similar to BART (Lewis et al., 2019). The model is trained to reconstruct the original input sequence from a corrupted version (*denoising*). Sequences are corrupted by randomly replacing text spans with a single mask token (*text infilling*) and permuting sentences (*sequence permutation*). The corrupted sequence is encoded using the bidirectional encoder, and the autoregressive decoder is trained to reconstruct the original sequence. The final loss is expressed as follows:

$$\mathcal{L} = \mathcal{L}_{NTPP} + \mathcal{L}_{Denoising}. \tag{5}$$

We refer to the overall model as BART+Layout2Pos.

Inference for Information Extraction Tasks To determine how the model predicts the next token of the sequence for information extraction tasks, we employ a customized variant of beam search to generate tokens while minimizing repetitions, therefore enhancing the coherence of the generated sequences. In this modified version, the generated tokens, if present in the source sequence, are constrained not to occur more frequently than in the original source. This constraint is enforced by keeping count of the number of occurrences of each token in the source sequence within the target sequence, masking the corresponding logit when the maximum occurrence is reached and redistributing the probability mass over the valid tokens.

5 Experiments

For reproducibility purposes, we will make the models implementation, along with the fine-tuning and evaluation scripts, publicly available.

5.1 Data

Pre-training Data Following a common practice in the field of Document Understanding, we collect data from the IIT-CDIP collection (Lewis et al., 2006) to build our pre-training dataset. IIT-CDIP consists of around 11 million scanned document page images of various types and layouts, including news articles, scientific reports, handwritten materials, and more. We select over 7 million document images from the collection to build our pre-training dataset, allocating over 18k for validation, another 18k for testing, and the remaining images for training. To extract text and bounding boxes from the

[3] It is noteworthy that a global reading order is unnecessary; there is no requirement to establish an order between two words that belong to segments that have no relation to each other.

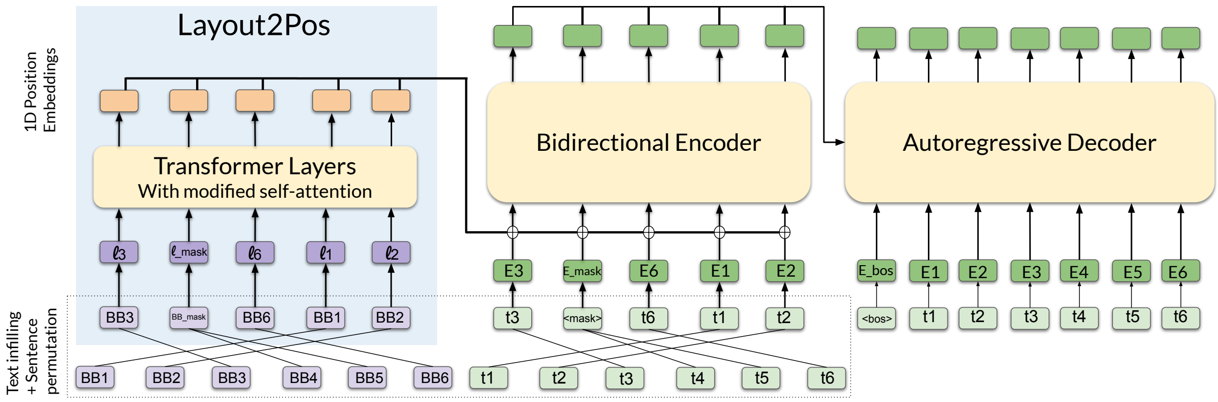


Figure 2: Architecture of Layout2Pos integrated into a BART model, *i.e.*, BART+Layout2Pos. The input consists of two components: a sequence of tokens and the corresponding sequence of token bounding box coordinates.

documents, we use DocTR (Mindee, 2021). Due to potential serialization errors induced by DocTR, and given that the Next Token Position Prediction task requires documents with proper reading orders, IIT-CDIP is only used for training models in language modeling tasks (*i.e.*, denoising).

To enable Layout2Pos to effectively learn the correct reading order of documents, we use the 500k documents from ReadingBank (Wang et al., 2021).[4] These documents are serialized and annotated with high-quality reading order annotations, serving as the training data for both Next Token Position Prediction and language modeling tasks.

In cases where a word is split into multiple tokens, approaches based on word-level bounding boxes typically assign the word's bounding box to all the tokens within that word. However, this approach is inefficient for Next Token Position Prediction, given that tokens within the same word would share identical layout embeddings, hence hindering accurate predictions of the next token. Therefore, we approximate token-level bounding boxes by dividing each word-level bounding box by the number of characters in the word.

Data for Visual Information Extraction We evaluate our approach on visual information extraction tasks, where the goal is to extract semantic entities from VrDs, based on a set of pre-defined keys. This evaluation is conducted using three benchmark for visual information extraction, each covering different document types: FUNSD (Jaume et al., 2019), SROIE (Huang et al., 2019), and CORD (Park et al., 2019). For each dataset, we use the reading order provided. To maintain consistency with pre-training data, we employ approximated token-level bounding boxes. We provide examples in Section C of the Appendix.

FUNSD (Jaume et al., 2019) is a form understanding dataset with 199 real, noisy and scanned forms where each sample is a list of form entities. There are three keys for which values have to be extracted: *question*, *answer*, and *header*. The dataset is split into 149 samples for training and 50 for test.

SROIE (v1) (Huang et al., 2019) is another receipt understanding dataset comprising 973 scanned receipts written in English. The task involves extracting entities for four keys: *total*, *date*, *company*, and *address*. The dataset is partitioned into 626 samples for training and 347 for test.

CORD (v1) (Park et al., 2019) is a receipt understanding dataset containing 1,000 scanned Indonesian receipts with 30 keys categorized into four superclasses: *menu*, *subtotal*, *total*, and *void*. Following the katanaml/cord[5] dataset repository, we exclude keys with very few occurrences, resulting in 22 keys grouped into three superclasses. The dataset is divided into 800 examples for training, 100 for validation, and 100 for test.

5.2 Experimental Settings

For full implementation details, see Section D of the Appendix.

Baselines We compare our approach with BART+2D, a layout-augmented BART model which relies on position embeddings derived from OCR-induced positions. These position embeddings are calculated using embedding tables and are subsequently added to textual features. Layout

[4] For further insights into the validation of this choice, see Section E of the Appendix.

[5] https://huggingface.co/datasets/katanaml/cord

embeddings, computed from bounding boxes using Equation 1, are incorporated to the resulting embeddings to construct the input embeddings. Following LayoutLMv2, BART+2D encodes spatial relative positions as bias terms added to the attention scores. BART+2D follows the same training and inference procedures as BART+Layout2Pos.

Additionally, we report the performance of two layout-aware encoder-only models: 1) LayoutLM and 2) LayoutLMv2-no-visual, a variant of LayoutLMv2 that discards visual information to ensure a fair comparison with our approach.

Pre-training Layout2Pos is composed of 2 layers with 12 attention heads and a hidden size of 768.[6] The final attention calculation, responsible for computing the next token matrix, involves a single attention head. Following the BART base model, both the encoder and decoder in BART+Layout2Pos and BART+2D are comprised of 6 layers, each with 12 attention heads and a hidden size of 768. BART+Layout2Pos comprises a total of 156M parameters, whereas BART+2D consists of approximately 140M parameters. Both models are trained from scratch on IIT-CDIP+ReadingBank for 10 epochs. We use a maximum sequence length of 512.

For LayoutLM, we use the microsoft/layoutlm-base-uncased checkpoint with 113M parameters, without any additional pre-training. Following the base architecture of LayoutLMv2, LayoutLMv2-no-visual is composed of a 12-layer Transformer encoder with 12 attention heads and a hidden size of 768, amounting to 110M parameters. The model is also pre-trained from scratch for 10 epochs on IIT-CDIP+ReadingBank, using Masked Visual Language Modeling (MVLM) (Xu et al., 2020b), a pre-training task that extends Masked Language Modeling (MLM) with layout information. The maximum sequence length is set to 512.

5.3 Visual Information Extraction

Sequence-labeling approaches We employ LayoutLM and LayoutLMv2-no-visual as sequence labeling methods. We use the BIO (Beginning, Inside, Outside) tagging format (Ramshaw and Marcus, 1999) as the labeling scheme to tag tokens based on both their entity and their position within that entity. For every dataset, the maximum sequence length is set to 512. Both models are fine-tuned for 100 epochs on FUNSD, and 20 epochs on SROIE and CORD.

Sequence-to-sequence models We frame visual information extraction as a sequence-to-sequence problem, wherein the document serves as the input, and the output consists of a series of extracted entities paired with their corresponding keys. For all three datasets, we set the maximum source sequence length to 512. Documents that exceed this length are split into contiguous sequences of 512 tokens each. For each input sequence, we formulate a target sequence containing the pairs of entities-keys to be extracted from the input sequence. The structure of the target sequences is defined such that each entity is followed by a colon and its corresponding key, with pairs separated by a line break. The arrangement of the pairs aligns with the order in which the corresponding entities appear in the document, *i.e.*, the provided reading order.

The pairs of generated and ground-truth *(entity, key)* are compared to compute precision, recall, and F1 score. To provide further insights into the model's errors, additional metrics are defined. To measure how often the model produces content that is not grounded in the input, the *hallucination rate* is defined as the percentage of entities generated by the model that do not match with any text in the input sequence. The *repetition rate* is the percentage of generated entities that are part of the ground-truth entities but are repeated more frequently than their occurrences in the ground-truth target sequence, quantifying the frequency with which the model repeats entities. The *wrong label rate* represents the proportion of generated entities present in the ground-truth but mislabeled by the model, and measures how often the model generates the right entities but mislabels them. The *omission rate* denotes the proportion of ground-truth entities that were not generated by the model, providing insights into how often the model omits entities. Lastly, the *non-entity rate* is the percentage of generated entities that, in the ground-truth, correspond to the category "Other". This metric assesses the frequency with which the model categorizes a text as an entity when it should not be considered as such (discarding hallucinations).

6 Results and Discussion

Table 2 reports the performance of all four models on FUNSD, SROIE, and CORD. We find that our sequence-to-sequence models achieve perfor-

[6]For further information regarding the validation of this architecture, see Section E of the Appendix.

Dataset	Reading Order	Model	Prec.	Rec.	F1	Rate				
						Repetition	Hallucination	Wrong Label	Omission	Non-entity
FUNSD	Original	LayoutLM (Xu et al., 2020b)	75.91	80.54	78.16					
		LayoutLMv2-no-visual	78.58	81.49	80.01					
		BART+2D	83.74	86.55	**85.12**	2.67	1.32	45.76	39.06	1.19
		BART+Layout2Pos	80.62	80.10	80.36	2.56	5.50	22.88	57.11	3.96
	Shuffled	BART+2D	77.82	82.16	79.93	2.89	2.15	48.37	34.68	3.25
SROIE	Original	LayoutLM (Xu et al., 2020b)	90.74	93.95	92.32					
		LayoutLMv2-no-visual	93.20	93.88	93.54					
		BART+2D	93.46	93.73	**93.60**	0.00	0.29	0.00	18.11	2.64
		BART+Layout2Pos	93.20	93.80	93.50	0.00	0.58	0.29	17.03	3.43
	Shuffled	BART+2D	80.58	66.33	73.13	2.66	1.46	0.41	73.34	5.72
CORD	Original	LayoutLM (Xu et al., 2020b)	93.91	95.11	94.51					
		LayoutLMv2-no-visual	93.14	94.89	94.00					
		BART+2D	95.97	94.81	**95.39**	2.33	0.00	5.28	19.06	0.33
		BART+Layout2Pos	94.56	92.71	93.62	0.99	4.37	5.40	22.83	0.40
	Shuffled	BART+2D	91.46	87.54	89.46	4.53	0.83	26.5	35.72	0.42

Table 2: Model performance (in %) on FUNSD, SROIE, and CORD, reported for 1) the original reading order and 2) three shuffled orders (averaged). Best F1 scores for each dataset/reading order are reported in bold.

mance that is comparable or even superior to sequence-labeling approaches. This suggests that the sequence-to-sequence approach can match the effectiveness of traditional sequence labeling methods, offering an alternative that is not constrained by the document's content and reading order. On SROIE, BART+Layout2Pos performs on par with its counterpart fed with sequential position information, BART+2D. This suggests that Layout2Pos effectively leverages layout information to generate meaningful position embeddings on SROIE, implying that the reading order provided by OCR is no longer necessary. However, on the other two datasets, BART+Layout2Pos demonstrates lower performance than BART+2D, with a slight underperformance on CORD and a more notable disparity on FUNSD. Additionally, we find that the majority of errors arise from either omitted or mislabeled entities. Overall, both models rarely hallucinate, repeat entities, or identify a text as an entity when it should not be considered as such.

To measure the impact of reading order on models dependent on it, we evaluate BART+2D on test documents with shuffled reading orders. For every test dataset, the reading order of each document is shuffled such that words belonging to the same entity remain grouped together. This process is repeated three times, generating three shuffled test sets for every original test dataset. BART+2D, fine-tuned using the reading order provided by the dataset, is then evaluated on each of the shuffled test sets. The resulting scores are then averaged and reported in Table 2. Results show that altering the reading order, even while ensuring that words belonging to the same entities are kept together, leads to a significant performance decline. Specifically, there is a F1-score drop of 5.19 and 5.93 for FUNSD and CORD, respectively, and a notable decrease of 20.84 for SROIE. This highlights the significance of developing methods robust to variations in reading order.

7 Conclusion

To derive position embeddings solely from layout information and avoid reading order issues, we propose Layout2Pos—a Transformer-based module that learns the sequential relationships between tokens in a document. We conduct experiments on three benchmarks datasets for visual information extraction, demonstrating the effectiveness of our approach in leveraging layout information to produce meaningful position embeddings. Furthermore, we showcase the significant impact of variations in reading order on models that rely on sequential position information, encouraging research on reading order-independent methods for document understanding tasks.

8 Limitations

In our sequence-to-sequence approach, any arrangement of key-value pairs is deemed valid. However, language models trained with teacher forcing tend to favor a single correct output, potentially penalizing valid responses with different entity orders. For future work, we will investigate permutation invariant losses to foster robustness to variation in entity orders.

While our current model evaluations have provided valuable insights, they are limited by their focus on relatively simple datasets and documents of shorter length. Additionally, our analyses have been confined to English language texts. Recognizing these limitations, we plan to enhance the generalizability by including more complex datasets, particularly those featuring longer documents (Graliński et al., 2020).

References

Srikar Appalaraju, Bhavan Jasani, Bhargava Urala Kota, Yusheng Xie, and R Manmatha. 2021. Docformer: End-to-end transformer for document understanding. *arXiv preprint arXiv:2106.11539*.

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*.

Filip Graliński, Tomasz Stanisławek, Anna Wróblewska, Dawid Lipiński, Agnieszka Kaliska, Paulina Rosalska, Bartosz Topolski, and Przemysław Biecek. 2020. Kleister: A novel task for information extraction involving long documents with complex layout. *arXiv preprint arXiv:2003.02356*.

Zhangxuan Gu, Changhua Meng, Ke Wang, Jun Lan, Weiqiang Wang, Ming Gu, and Liqing Zhang. 2022. Xylayoutlm: Towards layout-aware multimodal networks for visually-rich document understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4583–4592.

Jaekyu Ha, Robert M Haralick, and Ihsin T Phillips. 1995. Recursive xy cut using bounding boxes of connected components. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 2, pages 952–955. IEEE.

Zheng Huang, Kai Chen, Jianhua He, Xiang Bai, Dimosthenis Karatzas, Shijian Lu, and CV Jawahar. 2019. Icdar2019 competition on scanned receipt ocr and information extraction. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1516–1520. IEEE.

Guillaume Jaume, Hazim Kemal Ekenel, and Jean-Philippe Thiran. 2019. Funsd: A dataset for form understanding in noisy scanned documents. In *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, volume 2, pages 1–6. IEEE.

Anthony Kay. 2007. Tesseract: An open-source optical character recognition engine. *Linux J.*, 2007(159):2.

Geewook Kim, Teakgyu Hong, Moonbin Yim, JeongYeon Nam, Jinyoung Park, Jinyeong Yim, Wonseok Hwang, Sangdoo Yun, Dongyoon Han, and Seunghyun Park. 2022. Ocr-free document understanding transformer. In *European Conference on Computer Vision*, pages 498–517. Springer.

David Lewis, Gady Agam, Shlomo Argamon, Ophir Frieder, D Grossman, and Jefferson Heard. 2006. Building a test collection for complex document information processing. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 665–666.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Mindee. 2021. doctr: Document text recognition. `https://github.com/mindee/doctr`.

Hiroki Nakayama. 2018. seqeval: A python framework for sequence labeling evaluation. Software available from https://github.com/chakki-works/seqeval.

Seunghyun Park, Seung Shin, Bado Lee, Junyeop Lee, Jaeheung Surh, Minjoon Seo, and Hwalsuk Lee. 2019. Cord: a consolidated receipt dataset for post-ocr parsing. In *Workshop on Document Intelligence at NeurIPS 2019*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Qiming Peng, Yinxu Pan, Wenjin Wang, Bin Luo, Zhenyu Zhang, Zhengjie Huang, Teng Hu, Weichong Yin, Yongfeng Chen, Yin Zhang, et al. 2022. Ernie-layout: Layout knowledge enhanced pre-training for visually-rich document understanding. *arXiv preprint arXiv:2210.06155*.

Rafał Powalski, Łukasz Borchmann, Dawid Jurkiewicz, Tomasz Dwojak, Michał Pietruszka, and Gabriela Pałka. 2021. Going full-tilt boogie on document understanding with text-image-layout transformer. *arXiv preprint arXiv:2102.09550*.

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.

Lance A Ramshaw and Mitchell P Marcus. 1999. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer.

Clément Sage, Alex Aussem, Véronique Eglin, Haytham Elghazel, and Jérémy Espinas. 2020. End-to-end extraction of structured information from business documents with pointer-generator networks. In *Proceedings of the fourth workshop on structured prediction for NLP*, pages 43–52.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Benjamin Townsend, Eamon Ito-Fisher, Lily Zhang, and Madison May. 2021. Doc2dict: Information extraction as text generation. *arXiv preprint arXiv:2105.07510*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Zilong Wang, Yiheng Xu, Lei Cui, Jingbo Shang, and Furu Wei. 2021. Layoutreader: Pre-training of text and layout for reading order detection.

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. 2020a. Layoutlmv2: Multi-modal pre-training for visually-rich document understanding. *arXiv preprint arXiv:2012.14740*.

Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. 2020b. 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.

Chong Zhang, Ya Guo, Yi Tu, Huan Chen, Jinyang Tang, Huijia Zhu, Qi Zhang, and Tao Gui. 2023. Reading order matters: Information extraction from visually-rich documents by token path prediction. *arXiv preprint arXiv:2310.11016*.

A Preliminary Experiments: OCR Serialization Errors

We categorize the 100 documents from the ReadingBank subset into four prevalent document layout types: *plain* layout, *lists*, *multicolumn* layout, and *tables*. We provide examples in Figure 3.

B Pre-training Data

IIT-CDIP is made available under the terms of a custom license,[7] while ReadingBank is protected by Apache 2.0 license.

C Data for Visual Information Extraction

FUNSD is licensed under a custom (non-commercial) license.[8] CORD is made available under the terms of the Creative Commons Attribution 4.0 International License. The license information for SROIE is currently unavailable or undisclosed.

In Figure 4, we provide an example of source documents from FUNSD, SROIE, and CORD, along with their corresponding target sequences.

D Implementation Details

Models were implemented in Python using PyTorch[9] (Paszke et al., 2017) and Hugging Face[10] (Wolf et al., 2019) librairies.

D.1 Pre-training

Experiments were ran using Nvidia Titan RTX with 25GB.

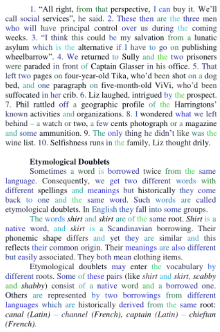
Pre-training Encoder-Decoder Models The documents are tokenized using the tokenizer of the base variant of BART (bart-base) shared through the Hugging Face Model Hub. The training spans 10 epochs, amounting to 500k optimization steps, including 59k steps for warmup. For each model, we select the checkpoint with the best validation loss. We use a maximum sequence length of 512, a batch size of 80, and a learning rate of $1e^{-4}$ which is linearly decayed. Following BART, we mask 30% of tokens in each sequence (with span lengths drawn from a Poisson distribution where $\lambda = 3$) and permute all sentences.

[7]https://www.industrydocuments.ucsf.edu/help/copyright/

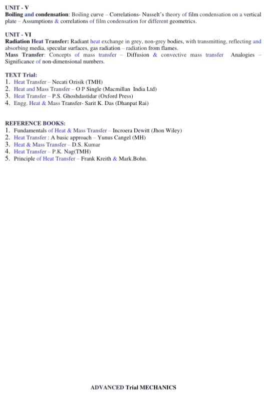
[8]https://guillaumejaume.github.io/FUNSD/work/

[9]Licensed under BSD 3-Clause License.

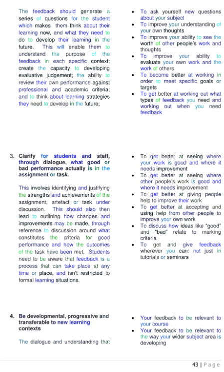
[10]Licensed under Apache License 2.0



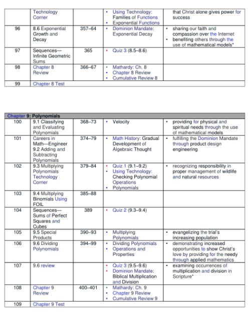
(a) Plain



(b) List

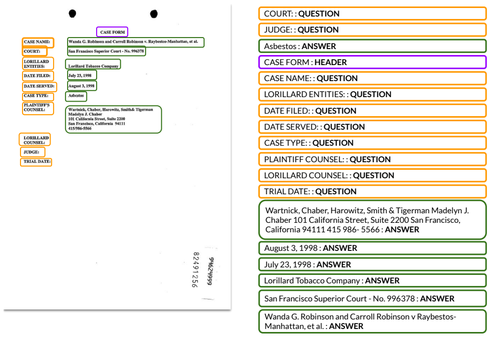


(c) Multicolumn

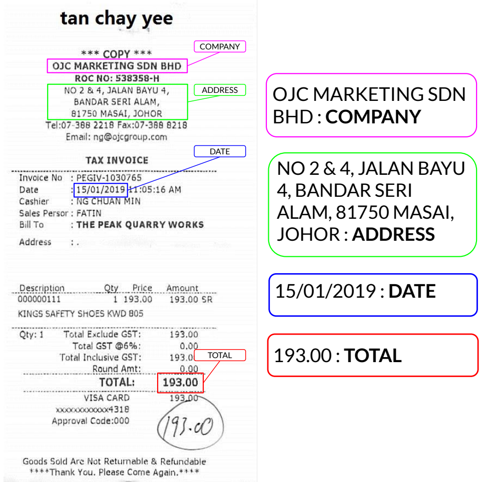


(d) Table

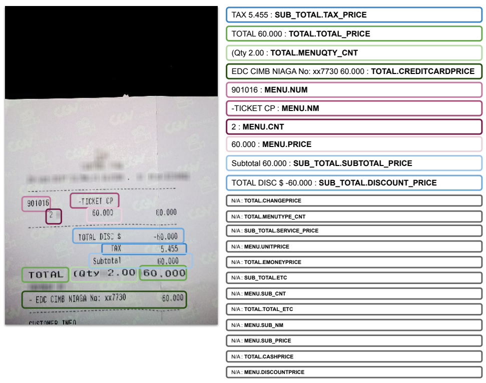
Figure 3: Examples of documents for each layout category, arranged from the simplest to the most complex.



(a) FUNSD



(b) SROIE



(c) CORD

Figure 4: Example document from FUNSD (a), SROIE (b), and CORD (c), accompanied by their corresponding target sequences that include the entities to be extracted paired with their corresponding keys. Best viewed in color.

Number of Layers	Pre-training Dataset	Accuracy (%)
1	IIT-CDIP	67.10
1	ReadingBank	89.37
2	ReadingBank	**95.86**

Table 3: Accuracy (in %) in predicting the next token for pairs sourced from ReadingBank, which were not used for pre-training. Selected pairs are considered "difficult", meaning that the tokens are positioned on different lines.

Pre-training Encoder-only Models The documents are tokenized using the tokenizer of microsoft/layoutlm-base-uncased. We use the Adam optimizer with weight decay fix (Loshchilov and Hutter, 2017), a weight decay of 0.01 and $(\beta_1, \beta_2) = (0.9, 0.999)$. We use a batch size of 80, and linear decay of the learning rate, which we set to $1e^{-4}$. Following BERT (Devlin et al., 2018), we mask 15% of the text tokens in MVLM, among which 80% are replaced by a special token `[MASK]`, 10% are replaced by a random token, and 10% remains the same.

D.2 Visual Information Extraction

Sequence-labeling approaches The learning rate is set to 5e-5 for both LayoutLM and LayoutLMv2-no-visual, on all datasets.

Sequence-to-sequence Models We compute statistics on the lengths of target sequences and establish the maximum target length to be greater than the 3rd quartile. In the case of FUNSD, we truncate target sequences at 768 tokens. As for SROIE and CORD, the maximum target sequence length is set to 96 and 512 tokens, respectively. BART+Layout2Pos (BART+2D) is fine-tuned for 100 (100), 40 (40), and 20 (50) epochs on FUNSD, SROIE, and CORD, respectively. The learning rate is set to 5e-5 for all models and datasets. During inference, we set the number of beams to 8. Precision, recall, and F1 scores are computed using the seqeval package (Nakayama, 2018).

E Next Token Position Prediction

We evaluate the performance of Layout2Pos by computing the accuracy of Next Token Position Prediction. This evaluation is conducted on a set of pairs of consecutive tokens derived from 100 examples from ReadingBank, which were not used in the pre-training phase. Specifically, we curated pairs categorized as "difficult", where the tokens are positioned on different lines, making a raster-scan approach ineffective. This choice demands the model to leverage layout information to accurately predict the next token in these scenarios.

In these experiments, we exclusively train and evaluate Layout2Pos, omitting the encoder-decoder architecture. For each token, we compute accuracy by comparing the position of its subsequent token with the position of the token associated with the highest logit according to Layout2Pos. We vary the number of layers and the pre-training dataset used.

Performance is reported in Table 3. Notably, pre-training Layout2Pos on ReadingBank compared to IIT-CDIP yields an increase of over 22% in accuracy. Additionally, our results indicate that augmenting the number of layers in Layout2Pos results in a notable increase of over 6% in accuracy. These results highlight the significance of using documents with accurate reading orders and contextualizing layout information to create position embeddings able to capture the reading order of documents.