DocTr: Document Transformer for Structured Information Extraction in Documents

Haofu Liao\(^1\)* Aruni RoyChowdhury\(^2\)† Weijian Li\(^3\)† Ankan Bansal\(^1\) Yuting Zhang\(^1\)
Zhuowen Tu\(^1\) Ravi Kumar Satzoda\(^1\) R. Manmatha\(^1\) Vijay Mahadevan\(^1\)

\(^1\)AWS AI Labs  \(^2\)MathWorks  \(^3\)Amazon Physical Stores

Abstract

We present a new formulation for structured information extraction (SIE) from visually rich documents. We address the limitations of existing IOB tagging and graph-based formulations, which are either overly reliant on the correct ordering of input text or struggle with decoding a complex graph. Instead, motivated by anchor-based object detectors in computer vision, we represent an entity as an anchor word and a bounding box, and represent entity linking as the association between anchor words. This is more robust to text ordering, and maintains a compact graph for entity linking. The formulation motivates us to introduce 1) a Document Transformer (DocTr) that aims at detecting and associating entity bounding boxes in visually rich documents, and 2) a simple pre-training strategy that helps learn entity detection in the context of language. Evaluations on three SIE benchmarks show the effectiveness of the proposed formulation, and the overall approach outperforms existing solutions.

1. Introduction

Structured information extraction (SIE) from documents, as shown in Fig 1, is the process of extracting entities and their relationships, and returning them in a structured format. Structured information in a document is usually visually-rich—it is not only determined by the content of text but also the layout, typesetting, and/or figures and tables present in the document. Therefore, unlike the traditional information extraction task in natural language processing (NLP) \cite{8, 3, 30} where the input is plain text (usually with a given reading order), SIE assumes the image representation of a document is available, and a pre-built optical character recognition (OCR) system may provide the unstructured text (i.e., without proper reading order). This is a practical assumption for day-to-day processing of business documents, where the documents are usually stored as images or PDFs, and the structured information, such as key-value pairs or line items (see Fig. 2) from invoices and receipts, has been primarily obtained manually. This is time consuming and does not scale well. Hence, automating the document structured information extraction process with efficiency and accuracy is of great practical and scientific importance.

Structured information extraction is part of document intelligence \cite{5}, which focuses on the automatic reading, understanding, and analysis of documents. Early approaches to document intelligence usually address the problem purely...
from either a computer vision or an NLP perspective. The former takes the document as an image input and frames entity detection as object detection or instance segmentation [41, 31]. The latter takes only the textual content of a document as the input, and addresses the problem with NLP solutions, such as IOB tagging via transformers [14].

Recently, models have also been proposed to pre-train on large-scale document collections and apply them to a wide variety of downstream document intelligence problems [38, 11, 1, 20]. Such general-purpose models usually have the ability to make use of multi-modal inputs – text from OCR, layout in the form of text locations, and visual features from images, and pre-training enables them to understand the basic structure of documents. Therefore, general-purpose models have demonstrated significant improvements on multiple document intelligence tasks, such as entity extraction [11, 20], document image classification [38, 1], and document visual question and answering [39, 1].

For structured information extraction, existing general-purpose models rely on two broad approaches: 1) IOB tagging [29] based methods [38, 39, 20], and 2) graph based methods [11, 15]. Both of these approaches suffer from inherent limitations. IOB tagging relies on the correct “reading order” or serialization of text, which however is not given by the OCR. As shown in Fig. 1(a), the raster scan order of OCR text separates I–name and E–name. When there are multiple name entities, it could be non-trivial to know which I–name/E–name word belongs to which name entity. Graph-based methods (Fig. 1(b)) can result in complex graphs with many words in a document (i.e., many nodes in the graph). Therefore, decoding the entities and their relationships from the adjacency matrices is error-prone.

Given the limitations of existing work, we make the following contributions in this paper:

• We introduce a new formulation for SIE where we represent an entity as an anchor word along with a box, and regard the problem as an anchor word based entity detection and association problem (Fig 1 (c)). Thus, we extract entities via bounding boxes and do not depend on the reading order of input. We assign each entity with an anchor word, resulting in a compact graph of entity relations (e.g., the anchor word links in Fig 1 (c)), which facilitates decoding structured information.

• We develop a new model, called Document Transformer (DocTr), which combines a language model and visual object detector for joint vision-language document understanding. We note that the recognition of an anchor word is largely a language-dependent task, while the detection of entity boxes is a more vision-dependent task. Therefore, DocTr is an intuitive approach to target this problem under the proposed formulation.

• We propose a new pre-training task, called masked detection modeling (MDM), that matches our formulation and helps learn box prediction in the context of language. Our experimental results show that 1) the proposed formulation addresses SIE better than IOB tagging or graph-based solutions, 2) MDM is a more effective pre-training task, in particular when worked together with the new formulation, and 3) the overall approach outperforms existing solutions on three SIE tasks.

2. Related Work

General-purpose document understanding. General-purpose approaches aim to develop a backbone model for document understanding, which is then adapted to address downstream document understanding tasks. LayoutLM [38, 39, 13] is an early approach that pre-trains on a large-scale document dataset. It introduces masked vision-language modeling and layout information for document understanding pre-training. BROS [11] improves LayoutLM via better encoding of the spatial information and introducing a pre-training loss for understanding text blocks in 2D. DocFormer [1] introduces a new architecture and pre-training losses to better leverage text, vision and spatial information in an end-to-end fashion. FormNet [20] encodes neighborhood context for each token using graph convolutions and introduces an attention mechanism to address imperfect serialization. StrucText [23] proposes to extract multi-modal semantic features at both token level, word-segment level and/or entity level. Donut [19] proposes an OCR free solution that is pre-trained to predict document text from images. It is an encoder-decoder model that can directly decode the expected outputs as text for downstream tasks.

Structured information extraction (SIE). Early approaches [41, 18, 6] formulate the SIE problem as a computer vision problem to either segment or detect entities from documents. However, they cannot address linking of entities due to the limitation of the formulation. With the advent of transformers [34] and their success in NLP, more recent approaches [25, 44, 10] address SIE by incorporating layout/visual information with text inputs to transformers, and extract entities via a NLP formulation [29]. Other approaches [24, 36, 42] propose to regard the text inputs as the nodes in a graph and model the relationship of text inputs via graph neural networks. To extract the relationship between entities, SPADE [15] introduces a graph decoding scheme on learned pairwise affinities between extracted entities.

Table detection and recognition (TDR). TDR is the task of detecting and recognizing tabular structures from document images. Both SIE and TDR focus on returning information in a structured way from documents. However, unlike SIE where the spatial relationship of entities are unconstrained, TDR assumes a tabular structure of entities (i.e., table cells) and leverages this prior knowledge in the model design and post-processing [28, 45, 26]. Moreover,
SIE requires returning a semantic label for each entity which demands an understanding of the text, while TDR does not distinguish between types of table cells but focuses more on table layout. Therefore, the existing approaches [28, 45, 26] to TDR are vision-only approaches.

**TextVQA.** Given an input image, TextVQA aims to answer questions related to the text in image. Similar to SIE, existing TextVQA approaches [32, 12, 9, 2] employ multi-modal models that take both the OCR and image as inputs. However, for TextVQA, the answers are typically single entities. It can be challenging to address the problem with TextVQA if we aim to return multiple entities in a structured way, and if an image could have multiple of such structures.

**Scene graph generation (SGG).** Generating scene graphs can be regarded as a form of SIE for natural images. SGG methods [17, 37, 43, 40, 33] detect objects as the nodes of scene graphs, and construct edges of scene graphs by identifying the pairwise relationships between objects. This is similar to our formulation of SIE where we extract entities via anchor word guided object detection, and link entities by learning to output their pairwise affinities.

### 3. Approach

#### 3.1. Structured Information Extraction

**Problem Formulation.** Following prior work, we assume the input is the image of a document page, and a pre-built OCR system is applied to detect and recognize the words. The goal of a structured information extraction system for document understanding is to extract a set of grouped entities \( G = \{G_i\} \), where each entity group \( G_i = \{e_{ij}\} \) is a set of entities with predefined relations. As shown in Fig. 2, an entity group may be a key and value pair, or a line item containing the name, count and price entities. We denote an entity as \( e = (t, c, b) \) where \( t, c \) and \( b \) are the text, class label, and location (bounding box) of the entity, respectively. Note that, with OCR inputs, this formulation of an entity can be reduced to \( e = (c, b) \), because the text \( t \) can be obtained by aggregating the OCR text inside \( b \).

Next, we propose a new formulation to address structured information extraction. We propose to address entity extraction via anchor word guided detection and entity linking via anchor word association. The former extracts entities \( \{e_i\} \), and the latter links entities into groups \( \{G_i\} \).

**Entity Extraction via Anchor Word Guided Detection.** To extract an entity \( e \), we first introduce a new concept called anchor word, which is a designated word of an entity. In Fig. 2, we select the first word of an entity as the anchor word, e.g., “ABC” is the anchor word for value, and “Chicken” is the anchor word for name. Other designations of anchor words are possible (see Sec. 4.2). An anchor word may be regarded as the representation of an entity. Since the goal of extracting an entity \( e = (c, b) \) is to find its class label \( c \) and bounding box \( b \), they may then be represented by an anchor word. As shown in Fig. 2, we associate each anchor word with a label and a bounding box. For example, the anchor word “Ship” is associated with a label key and a bounding box that encloses the entity “Ship To:”. Therefore, the task of extracting an entity may be seen as first identifying its anchor word, and then obtaining the label and bounding box associated with it.

**Entity Linking via Anchor Word Association.** We define an entity group as consisting of a primary entity, and all the other entities in the group are secondary. The anchor word of a primary/secondary entity is the primary/secondary anchor word. Once anchor words have been identified, linking entities into group is equivalent to associating anchor words. To establish such association, we first select the primary anchor words of entity groups, and then all the secondary anchor words from the same group are linked to the primary anchor word. The definition of a primary entity may vary. For key-value pairs, the primary anchor words may simply be those anchor words labeled as key. For more general entity groups, we designate a primary anchor word based on the task/data. For example, we choose name’s anchor word “Chicken” as the primary anchor in Fig. 2. Other ways of choosing primary anchors are possible (See Sec. 4.2). Links between primary and secondary anchor words are represented by a binary matrix \( M \in \{0, 1\}^{m \times n} \). \( M_{ij} = 1 \) indicates that the \( i \)th primary anchor word, and \( j \)th second anchor word are linked. Otherwise, \( M_{ij} = 0 \).

#### 3.2. DocTr: Document Transformer

DocTr is a multi-modal transformer that takes both the document image and OCR words (text and position) as input. Unlike existing encoder-only approaches [39, 1, 23], DocTr has an encoder-decoder architecture with 1) two dedicated encoders to encode vision and language features separately,
2D position embeddings of OCR) along with the OCR text as input. However, no visual information is added since it has already been addressed by the vision encoder. The language encoder is critical to our formulation for the identification of anchor words, which is a language-dependent task.

Vision-Language Decoder with Language-Conditioned Queries. The architecture of the vision-language decoder is similar to the decoder of the Deformable DETR transformer model [46] - with two major differences to facilitate the decoding of vision-language inputs. Each decoder layer has two cross-attention modules to decode from vision and language inputs respectively. For vision, we apply deformable cross-attention (similar to Deformable DETR) to efficiently decode from high-resolution visual features. For language, we apply language-conditioned cross-attention to decode from the discrete OCR language features.

Specifically, we introduce language-conditioned queries to better leverage the OCR inputs and obviate the need for bipartite matching between predicted and ground truth entities. The original DETR-like decoder queries [4, 46] do not have explicit semantic meanings at the beginning. Hence, DETR requires finding the most plausible matching between a prediction and ground truth, which is less effective and impedes the training. For document understanding with OCR inputs, we consider a one-to-one mapping between OCR inputs and decoder queries. That is, we have the same number of queries as the number of OCR inputs to the language encoder, and the ith query is mapped to the ith OCR input (see Fig. 3). This mapping can be simply modeled as cross-attention between queries and language embeddings by using the same position embedding for both inputs. Let $Q \in \mathbb{R}^{L \times d}$ be a set of L decoder queries each with dimension $d$ (packed as a matrix), $V \in \mathbb{R}^{L \times d}$ be the set of output embeddings from the language encoder, and $P \in \mathbb{R}^{L \times d}$ be a set of position embeddings. Then, the cross attention with language-conditioned queries can be written as:

$$\text{CrossAttn}(Q, V, P) = \text{softmax}\left(\frac{(Q+P)(V+P)^T}{\sqrt{d}}\right)V, \quad (1)$$

where $\sqrt{d}$ is a scaling factor [34]. This mapping assigns each query with an explicit linguistic semantic meaning – the i-th decoder output now corresponds to the i-th input text token, via the i-th decoder query. Thus, we can directly match entities with queries without the bipartite matching required by the default DETR decoder formulation [4, 46].

Entity Extraction and Linking Outputs. The decoder has two sets of outputs for entity extraction and entity linking respectively (see Fig. 3). For entity extraction, each output is a class label and a bounding box which uniquely decide an entity. Because each query (and thus its corresponding output) is mapped to an OCR input, the class label indicates
whether the underlying OCR input is an anchor word, and
the type of entity it represents. For entity linking, each output
is a binary class label and an embedding vector. The binary
class label indicates whether the OCR input is a primary
anchor word. The embedding vector is for the linking of
anchor words, and we use different embeddings for primary
and secondary anchor words. Let \( E_p \in \mathbb{R}^{m \times h} \) be a set of \( m \)
primary embeddings, and \( E_s \in \mathbb{R}^{n \times h} \) be a set of \( n \) secondary
embeddings, the predicted affinity matrix for entity linking
is computed as \( \hat{M} = \text{sigmoid}(E_p E_s^T) \), \( \hat{M} \in \{0, 1\}^{m \times n} \).

### 3.3. Architecture Details

For the vision encoder, we use a ResNet50 backbone and
a 6-layer deformable transformer encoder [46]. The back-
bone is initialized with ImageNet pretrained weights, and
outputs three scales of visual features. The multi-scale
visual features are transformed into a sequence with 2D “sine”
position embeddings before sending to the deformable
transformer encoder. For the language encoder, we use a 12-layer
transformer encoder with the same architecture settings as
the BERT-base model [7]. In addition to BERT’s text embed-
dings and 1D position embeddings, we also add 2D position
embeddings [38] to include layout information of the doc-
ument as the input. The 2D position embeddings are learned
embeddings with random initialization. The VL-decoder has
6 layers, where each layer consists of a self-attention mod-
ule, a deformable cross-attention module [46] and a standard
cross-attention module [33] (see supplementary material for
detailed architecture of VL-decoder layers).

### 3.4. Training and Pre-training

#### Entity Extraction and Linking Objectives.

The entity extraction objective is similar to the one used in DETR [4]
except that we do not need the bipartite matching due to
the use of language-conditioned queries (as introduced in
Sec. 3.2). Specifically, given a set of \( N \) OCR inputs, the
language-conditioned queries yields \( N \) entity extraction out-
puts \( \hat{E} = \{\hat{e}_i\}_{i=1}^N \). For a document with \( M \) entities, we
also construct a ground truth \( E = \{e_i\}_{i=1}^N \) of size \( N \). Here,
\( \hat{e}_i \) and \( e_i \) denote the predicted and ground truth entities of
the \( i \)th OCR input, respectively. Note that not every OCR
word is an anchor word, and thus it may have no associated
time. In this case, we say that the ground truth of the input
OCR is an empty entity, i.e., \( e = \emptyset \), and there are in total
\( N - M \) empty entities in \( E \). If we denote a non-empty entity as
\( e = (c, b) \) and a predicted entity as \( \hat{e} = (\hat{p}, \hat{b}) \), where \( c \) is
the ground truth entity label, \( \hat{p} \) is the predicted entity label
probability, and \( \hat{b} / \emptyset \) is the ground truth/predicted bounding
box, then we write the entity extraction loss as

\[
\mathcal{L}_{\text{EE}}(E, \hat{E}) = \sum_i [-\log \hat{p}_i(c_i) + \lambda \mathbb{I}_{\{e_i \neq \emptyset\}} \mathcal{L}_{\text{bbox}}(b_i, \hat{b}_i)],
\]

(2)

where \( \hat{p}_i(c_i) \) is the predicted probability of entity being la-
belled as \( c_i \), \( \mathcal{L}_{\text{bbox}} \) is a bounding box loss [4], and \( \mathbb{I}_{\{e_i \neq \emptyset\}} \)
means we only compute \( \mathcal{L}_{\text{bbox}} \) for non-empty entities.

The entity linking loss consists of two parts, primary
anchor classification and linking classification. Let \( \hat{\mathbf{L}} \) be
a set of primary anchor classification outputs and \( \mathbf{L} \) be its
ground truth binary labels. Let \( \hat{\mathbf{M}} \) and \( \mathbf{M} \) be the predicted
ground truth entity linking affinity matrices, respectively.
Then, we can simply write the entity linking loss as

\[
\mathcal{L}_{\text{EL}}(\mathbf{L}, \hat{\mathbf{L}}, \mathbf{M}, \hat{\mathbf{M}}) = \text{BCE}(\mathbf{L}, \hat{\mathbf{L}}) + \beta \text{BCE}(\mathbf{M}, \hat{\mathbf{M}}),
\]

(3)

where BCE denotes the binary cross-entropy loss.

#### Pre-training.

We pre-train DocTr on a large-scale dataset
of unlabeled document images. For simplicity of modeling,
we only include one pre-training task, termed masked
detection modeling (MDM), for DocTr which we find suffi-
cient for downstream tasks. Since pre-training is not the
main focus of this work, we leave the exploration of other
pre-training strategies [39, 11, 1] for future work. Fig. 4
illustrates MDM and compares it with related pre-training
tasks. MDM is an extension of masked vision-language
modeling (MVLM) [38, 39]. Both MDM and MVLM take OCR
text and boxes as input. However, MVLM only randomly
masks the text inputs. Instead, MDM randomly masks both
the text inputs and their boxes. Specifically, we replace text
with [MASK] and set boxes to [0, 0, 0, 0]. Then, we
train DocTr to predict both the masked texts and their cor-
responding boxes. Note that this task is similar to object
detection. Thus, the objective function can be written in
the same way as Eq. (2), where the first term is for masked
text classification, and the second term is for masked box
regression. Also note that for MDM, the input image is not
masked so that a model can better learn how to leverage the
visual information to locate and identify the masked inputs.
Datasets and Tasks. We use three datasets in our experiments, IIT-CDIP document collection [21], CORD [27] and FUNSD [16]. We follow the convention in the literature [38, 39, 11, 1] to pre-train DocTr on the IIT-CDIP document collection, which is a large-scale dataset with 11 million unlabeled documents. CORD [27] is a receipt dataset with 800 training, 100 validation, and 100 testing samples. Each receipt in this dataset is labeled with a list of line items and key-value pairs. FUNSD [16] consists of scanned forms, with 149 training and 50 testing examples. Each form is labeled with key/value entities together with links to indicate which keys and values are associated.

We evaluate our model’s performance on three tasks, receipt parsing, entity labeling and entity linking. For receipt parsing, a model not only has to extract each receipt’s entities but also correctly link entities to form line items and key-value pair groups. Fig. 5 (a) shows a sample receipt from CORD and its expected output after parsing. The sample contains two line items and four key-value pairs. For line items, it requires identifying each line item related entity (class and text) and group the entities of the same line item together. For key-value pairs, we identify class labels of the keys and return only text of the corresponding values. We use the same evaluation protocols and metrics as defined in [15] to evaluate the receipt parsing performance.

Entity labeling and entity linking are commonly adopted tasks [38, 15] to evaluate a pre-trained model’s performance, which however are simplified versions of what we have defined in Sec. 3.1. Entity labeling requires assigning a class label to each word of the document. Fig. 5 (b) shows a sample from FUNSD where the task is to identify if a word belongs to a key (red), a value (green) or a title (blue). In entity linking, the assumption is that the key/value entities are correctly detected, and the task is to identify which keys and values should be linked (See Fig. 5 (c), red arrows). We evaluate entity labeling/linking by checking if the words/links are correctly labeled using F1-score as the metric.

4. Experiments

4.1. Comparison with Existing Solutions

We compare DocTr with the existing methods on receipts parsing, entity labeling, and entity linking tasks, respectively.

For receipts parsing, SPADE [15] and Donut [19] are the only two other publicly available solutions (to the best of our knowledge) that address this task on CORD. The other existing general-purpose models [39, 11, 13] are not able to directly address this structured information extraction task out-of-the-box. For a fair comparison with our method, we fine-tune the officially released general-purpose models under two settings: using the standard IOB tagging for receipts parsing or using our proposed formulation. From Table 1, we can see that DocTr outperforms general-purpose models BROS, LayoutLMv2 and LayoutLMv3 by a noticeable margin when they are fine-tuned with the IOB tagging setting. When fine-tuned with our proposed formulation, the general-purpose models’ performance improved but they are still behind DocTr, which shows the effectiveness of the proposed encoder-decoder solution for the anchor word based structure information extraction.

For entity labeling, we note that the majority of existing works (including DocTr) report numbers using word-level boxes (for position embedding) as input, and some others [22, 35, 13] use segment-level boxes from GT as input. Segment-level boxes provide more semantic information, and thus their usage is unfair to those using word-level boxes. Since segment-level boxes are not conventional inputs (due to OCR limitations), in this experiments we mainly focus on comparing the methods with word-level boxes.

To address entity labeling, DocTr follows the general-
Table 3: Comparison with existing solutions on entity labeling (with FUNSD and CORD datasets).

<table>
<thead>
<tr>
<th>model</th>
<th>text box</th>
<th>FUNSD</th>
<th>CORD</th>
<th>#params</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPADE [15]</td>
<td>word</td>
<td>71.6</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LayoutLM BASE [38]</td>
<td>word</td>
<td>78.7</td>
<td>94.7</td>
<td>113M</td>
</tr>
<tr>
<td>BROS BASE [11]</td>
<td>word</td>
<td>83.1</td>
<td>96.5</td>
<td>110M</td>
</tr>
<tr>
<td>DocFormer BASE [11]</td>
<td>word</td>
<td>83.3</td>
<td>96.3</td>
<td>183M</td>
</tr>
<tr>
<td>LayoutLM2 BASE [39]</td>
<td>word</td>
<td>82.8</td>
<td>95.0</td>
<td>200M</td>
</tr>
<tr>
<td>StructText [23]</td>
<td>word</td>
<td>83.4</td>
<td>-</td>
<td>107M</td>
</tr>
<tr>
<td>DocTr (ours)</td>
<td>word</td>
<td>84.0</td>
<td>98.2</td>
<td>153M</td>
</tr>
<tr>
<td>LayoutLM LARGE [38]</td>
<td>word</td>
<td>79.0</td>
<td>95.0</td>
<td>343M</td>
</tr>
<tr>
<td>BROS LARGE [11]</td>
<td>word</td>
<td>84.5</td>
<td>97.3</td>
<td>340M</td>
</tr>
<tr>
<td>DocFormer LARGE [11]</td>
<td>word</td>
<td>84.5</td>
<td>97.0</td>
<td>536M</td>
</tr>
<tr>
<td>LayoutLM2 LARGE [39]</td>
<td>word</td>
<td>84.2</td>
<td>96.0</td>
<td>426M</td>
</tr>
<tr>
<td>FormNet [20]</td>
<td>word</td>
<td>84.7</td>
<td>97.3</td>
<td>345M</td>
</tr>
<tr>
<td>LiLT BASE [35]</td>
<td>segment</td>
<td>88.4</td>
<td>96.1</td>
<td>-</td>
</tr>
<tr>
<td>LayoutLM3 BASE [13]</td>
<td>segment</td>
<td>90.3</td>
<td>96.6</td>
<td>133M</td>
</tr>
<tr>
<td>StructualLM LARGE [22]</td>
<td>segment</td>
<td>85.1</td>
<td>-</td>
<td>426M</td>
</tr>
<tr>
<td>LayoutLM3 LARGE [13]</td>
<td>segment</td>
<td>92.1</td>
<td>97.5</td>
<td>368M</td>
</tr>
</tbody>
</table>

Table 4: Comparison of different SIE formulations under two text serialization settings, raster scan and oracle.

<table>
<thead>
<tr>
<th>formulation</th>
<th>text serial.</th>
<th>parsing (C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOB tagging [29]</td>
<td>raster scan</td>
<td>93.2</td>
</tr>
<tr>
<td>SPADE [15]</td>
<td>raster scan</td>
<td>93.0</td>
</tr>
<tr>
<td>DocTr (ours)</td>
<td>raster scan</td>
<td>94.4</td>
</tr>
<tr>
<td>IOB tagging [29]</td>
<td>oracle</td>
<td>94.1</td>
</tr>
<tr>
<td>SPADE [15]</td>
<td>oracle</td>
<td>93.9</td>
</tr>
<tr>
<td>DocTr (ours)</td>
<td>oracle</td>
<td>95.0</td>
</tr>
</tbody>
</table>

Figure 6: Visualization of receipt parsing results using different SIE formulations. Each result consists of the visualization of model predictions, and the parsing outputs (given the model predictions).

(a) IOB tagging visualizes the predicted tags of OCR words. (b) SPADE decoding visualizes the decoded graph, and arrows between words indicate that the words are linked in the same entity. (c) DocTr visualizes the predicted anchor words and their bounding boxes. For the parsing outputs, green/red text means the predicted text matches/does not match ground truth. Strikethrough text means the ground truth text is missed from prediction.

OCR words) and apply different formulations to decode structured information. Specifically, we compare our formulation with IOB tagging and graph based solutions. For IOB tagging, we follow the literature [20, 38] and assign BIOES tags to each token and decode entities according to the tagged entity spans. Note that IOB tagging does not support entity linking. For a fair comparison, we link entities using a way similar to the anchor word association method introduced in Sec. 3.2. We treat the “B” tag or “S” tag of entities as the anchor words and link entities via decoding of entity linking affinity matrices. For graph based SIE, we follow the literature [11, 15] by attaching a SPADE [15] decoder at the end of DocTr. We fine-tune DocTr and decode graphs using the same way as specified in the original SPADE method. To understand the sensitivity of the SIE formulations with regard to the reading orders of input text, we evaluate them under two text serialization settings, raster scan and oracle. For oracle, we first order the ground truth entities in a raster scan manner, then order text while preserving the entity order.

Table 4 shows the receipt parsing results on the CORD dataset. Our proposed formulation achieves the best performance in both text serialization settings. We notice that, compared with the other two formulations, our formulation is less sensitive to text serialization with only 0.6 score drop (vs. 0.9 drop by IOB tagging or SPADE) while switching

4.2. Model Properties

We analyze DocTr’s design and consider other choices.

Problem Formulation. We use DocTr as the backbone network for the encoding of document inputs (image and...
from oracle to raster scan text serialization. We also observe that our formulation can better address cases where there is dense text with multiple entities near each other. Fig. 6 shows an example visualization (see supplementary material for more results). For IOB tagging, it can tag most of the words well. However, even a single tagging error can cause failures of entity decoding, and an entity is missed from the parsing outputs. For SPADE, the dense words result in a challenge for constructing an entity graph, and the model incorrectly merges the two sub_nms’s as a single entity. In comparison, DocTr only requires identifying the anchor words which is an easier task and, with bounding box predictions, all the entities are correctly extracted.

### Anchor Word and Primary Anchor

We investigate different ways of designating anchor word and primary anchor. In Sec. 3.1, we introduced using the first word (in terms of reading order) of an entity as the anchor word. Here, we consider two alternatives: 1) using the last word or 2) both the first and last word as the anchor. Table 5 (row 1-2, 4) shows the comparison of these three choices. We notice that there is no significant differences (94.2 vs. 94.1) between using the first word and last word as the anchor word. Using both first and last as the anchor word gives slightly better performance. We hypothesize that this is because first and last words help better identify the boundary of an entity.

For primary anchor, we investigate its choices for line-item extraction. We consider two candidates: 1) using the anchor word of the first entity in a line-item, or 2) using the anchor word of name as the primary anchor. From Table 5 (row 3 and 4), we see the latter is a better choice with 0.4 improvement. This is reasonable since the first entity in a line-item may vary semantically (i.e., it could be name, cnt or other entity types), and thus it is harder to identify. However, this choice is also more flexible than using name as the primary anchor because there may be no name in an line-item. For CORD, each line-item always has a name, so this is not a concern (see supplementary material for primary anchor choices of other entity categories).

### Pre-training

We evaluate the effectiveness of the pre-training task (MDM) introduced in Sec. 3.4. We consider three settings: 1) without pre-training, 2) with MVLM and 3) with MDM. Table 6 compares their performances. Without pre-training, the performance drops significantly. With MVLM, the performance improves but still falls behind using MDM. This shows the effectiveness of having MDM for document understanding pre-training. In particular, we see more benefit of using MDM for receipt parsing. This is because our proposed formulation requires bounding box regression, and MDM helps learn better box predictions.

We also show example MDM pre-training predictions in Figure 7. Note that since both the input OCR box and text are masked, the model will need to not only predict what is masked but also predict where to find the masked word. We can see in most of the cases, the model can predict both kinds of information well. There are cases where the box (e.g., “25.000” in row 1) or text (e.g., “Bun” in row 2) is not accurately predicted. But the errors are reasonable. We also notice that the model can predict words that cannot be inferred through only text context, such as prices. This shows the usage of visual information.

### Table 5: Receipt parsing (CORD) results under different choices of anchor words and primary anchors.

<table>
<thead>
<tr>
<th>Anchor Word</th>
<th>Primary Anchor</th>
<th>Parsing (C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>first</td>
<td>name</td>
<td>94.2</td>
</tr>
<tr>
<td>last</td>
<td>name</td>
<td>94.1</td>
</tr>
<tr>
<td>first + last</td>
<td>first</td>
<td>94.0</td>
</tr>
<tr>
<td>first + last</td>
<td>name</td>
<td>94.4</td>
</tr>
</tbody>
</table>

### Table 6: Receipt parsing (CORD), entity labeling (FUNSD) and entity linking (FUNSD) results using different pre-training tasks.

<table>
<thead>
<tr>
<th>Pre-training</th>
<th>Parsing (C)</th>
<th>ELB (F)</th>
<th>ELK (F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>82.3</td>
<td>14.2</td>
<td>12.0</td>
</tr>
<tr>
<td>MVLM [38]</td>
<td>90.9</td>
<td>82.7</td>
<td>73.0</td>
</tr>
<tr>
<td>MDM</td>
<td>94.4</td>
<td>84.0</td>
<td>73.9</td>
</tr>
</tbody>
</table>

### Figure 7: Example pre-training predictions on CORD images. For inputs, we visualize masked word boxes, and their text is replace by [MASK]. For predictions, we visualize the predicted word boxes of the masked inputs. Under each box prediction, we also visualize its corresponding word token predictions.
**Architecture Design.** We then ablate the impact of the architectural components of DocTr. Table 7 shows the ablation results. The first row is a DocTr model with only the language encoder which is equivalent to the LayoutLM [38] model without visual inputs. The second row is a model with both the language encoder and VL-decoder but no vision encoder. These two models are close in performance. This is reasonable as without visual inputs the VL-decoder does not add much of information for decoding. Row 4 is the full model with both the vision encoder and VL-decoder. Compared with row 1 and 2, the performance improves noticeably. This suggests the importance of using visual information.

For row 3 and 4, we study the effectiveness of using the proposed language conditioned queries (LCQ). Specifically, we apply Eq. (1) to the cross-attention module when LCQ is checked. Otherwise, the standard cross-attention is used. We can see that LCQ is important since it helps to guide this one-to-one mapping between OCR and outputs, which is required by our proposed formulation.

**5. Conclusion**

We have presented a new approach for SIE from visually-rich documents. This approach is based on our novel formulation which includes object detection as part of the problem setting. This naturally leads us to include a transformer-based object detector as part of the architecture design and an object detection based loss in pre-training.

We have empirically shown that our proposed object detection based formulation readily addresses the structured information extraction task, and our solution outperforms existing solutions on SIE benchmarks. We hope this approach will initiate more efforts in combining object detection with existing vision-language models for document intelligence.

We also note that using anchor words limits the application of this approach to text-rich documents, and text-based entity extraction only. For future work, we explore solutions that extend the propose formullation for the extraction of non-contextual content (e.g., symbols, logos, etc.) from documents.

**References**


[38] Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. Layoutlm: Pre-training of text and layout