LMDX: Language Model-based Document Information Extraction and Localization

Vincent Perot[1*], **Kai Kang**[2], **Florian Luisier**[2], **Guolong Su**[1], **Xiaoyu Sun**[2], **Ramya Sree Boppana**[2], **Zilong Wang**[5], **Zifeng Wang**[3], **Jiaqi Mu**[1], **Hao Zhang**[4], **Chen-Yu Lee**[3], **Nan Hua**[1]
[1]Google DeepMind [2]Google Cloud
[3]Google Cloud AI Research [4]Google [5]UC San Diego

Abstract

Large Language Models (LLM) have revolutionized Natural Language Processing (NLP), improving state-of-the-art and exhibiting emergent capabilities across various tasks. However, their application in extracting information from visually rich documents, which is at the core of many document processing workflows and involving the extraction of key entities from semi-structured documents, has not yet been successful. The main obstacles to adopting LLMs for this task include the absence of layout encoding within LLMs, which is critical for high quality extraction, and the lack of a grounding mechanism to localize the predicted entities within the document. In this paper, we introduce ***L**anguage **M**odel-based **D**ocument Information **EX**traction and Localization* (LMDX), a methodology to reframe the document information extraction task for a LLM. LMDX enables extraction of singular, repeated, and hierarchical entities, both with and without training data, while providing grounding guarantees and localizing the entities within the document. Finally, we apply LMDX to the PaLM 2-S and Gemini Pro LLMs and evaluate it on VRDU and CORD benchmarks, setting a new state-of-the-art and showing how LMDX enables the creation of high quality, data-efficient parsers.

1 Introduction

The recent advent of transformers (Vaswani et al., 2017) and self-supervised pretraining procedures has led to significant progress in Visually Rich Document (VRD) Understanding. Within that field, the task of document information extraction (IE), which consists of extracting key entities within a semi-structured document (e.g. invoice, tax form, paystub, receipt, etc) given a predefined schema, has received a lot of attention from industry and academia due to its importance and wide applicability to intelligent document processing workflows. However, document information extraction still remains challenging for mainstream systems.

In particular, information in semi-structured forms is organized in complex layout across many possible templates, which requires deep understanding of the document context, spatial alignment among the different segments of text, and tabular arrangement of hierarchical entities (we define hierarchical entities as entities that are composed of logically grouped leaf entities, e.g. line items on an invoice composed of item description, quantity and price, or deduction items on a paystub composed of deduction type and amount, etc.). Moreover, since some business document automation workflows require certain level of accuracy, they are often integrated with human-in-the-loop interactions for auditing and correction of predictions, requiring knowing the precise location of extracted entities to make it a tractable task for a human rater. Finally, since a quasi-infinite number of document types exist, and that organizations have limited annotation resources, most parsers are built with very small amount of training data. From those complexities emerge the following desiderata of document information extraction systems: they should **(1) support high-quality extraction of singular, repeated, and hierarchical entities**, while **(2) localizing those entities in the document**, and **(3) do so with very low or no data annotation cost**.

So far, no publicly disclosed system has been able to address all of those desiderata. Current mainstream document IE systems are based on sequence-tagging and sequence-generation. Sequence-tagging approaches (Palm et al., 2017; Lee et al., 2021, 2022, 2023a; Wang et al., 2023c) classifies each token into Inside-Outside-Begin (IOB) tags (Ramshaw and Marcus, 1995), which support extraction and localization of leaf entities. However, it is non-trivial to extend these meth-

[*] Correspondence to <vperot@google.com>.

Findings of the Association for Computational Linguistics: ACL 2024, pages 15140–15168

ods for hierarchical entities. Sequence-generation based methods (Powalski et al., 2021; Kim et al., 2022) treats extraction as text generation with auto-regressive decoders (Sutskever et al., 2014). Although this line of work allows to predict hierarchical entities, it does not allow localizing entities in the document. Moreover, both categories require significant human annotation cost to ensure a high quality extraction. Thus, a unified IE framework addressing all three desiderata is highly valuable.

In parallel, Large Language Models (LLMs) (OpenAI, 2023a; Google et al., 2023; Hoffmann et al., 2022) have revolutionized Natural Language Processing, showing the capabilities to solve diverse tasks with an instruction (Wei et al., 2022) or a few demonstrations attached to the prompt (Brown et al., 2020). This paradigm shift opens the possibility of extracting entities while addressing all the aforementioned desiderata, but using LLMs for VRD IE has been underexplored. Wang et al. (2023a) proposes a document-centric LLM and frames IE as a question-answer task, enabling zero-shot extraction but lacking support of localization and hierarchical entity extraction. Furthermore, this can suffer from hallucinations, a common issue with LLMs (Huang et al., 2023).

This motivates us to introduce ***L**anguage **M**odel-based **D**ocument Information E**X**traction and Localization* (LMDX), a principled methodology for leveraging existing LLMs for information extraction and localization on visually-rich documents, meeting all three identified desiderata of extraction systems and detailed in Figure 1. A comparison of LMDX characteristics and other document information extraction systems can be found at Table 1. Our contributions can be summarized as follows:

- We present a principled recipe that enables LLMs to perform the document IE task on leaf and hierarchical entities with precise entity localization, including without any training data, and using only the simple text-in, text-out interface that is applicable to LLMs.
- We propose a layout encoding scheme that communicates spatial information to the LLM without any changes to its architecture.
- We introduce a decoding algorithm that transforms the responses from the LLM into extracted entities and their corresponding bounding boxes on the document, while discarding any LLM hallucination.
- We systematically evaluate the data efficiency of LMDX across multiple public benchmarks, establishing a new state-of-the-art, and provide extensive study of the different core designs to demonstrate their effectiveness.

Table 1: Comparison of document information extraction systems. Unlike mainstream document IE systems, LMDX enables the zero-shot extraction, including hierarchical entities, all while localizing its predictions.

Document Information Extraction Systems	Hierarchical entity	Entity localization	Zero-shot support
LayoutLMv3 (Huang et al., 2022), FormNetV2 (Lee et al., 2023a),	✗	✓	✗
Donut (Kim et al., 2022)	✓	✗	✗
DocLLM (Wang et al., 2023a)	✗	✗	✓
LMDX (Ours)	✓	✓	✓

2 Related Work

Framing IE from VRD. Information Extraction from VRD is a complex task that can be framed in a variety of ways. Many approaches divide the problem in two sub-tasks: a text recognition/serialization step, typically achieved by an Optical Character Recognition (OCR) service, followed by a parsing step, which finds the relevant entity values from the recognized text. Xu et al. (2021); Appalaraju et al. (2021) frame this parsing step as Named Entity Recognition (NER), encoding each token with a transformer encoder and classifying each document token into IOB tags, allowing extraction and localization of leaf entities only. Other approaches treat extraction as a sequence generation problem. Powalski et al. (2021) adds an auto-regressive decoder on top of a text-layout-image encoder, all initialized from T5 (Raffel et al., 2020). This enables to predict hierarchical entities, but not localize entities in the document. While LMDX still frames VRD IE as a sequence generation task, our work contrasts with prior work by combining the advantages of the different framings as shown in Table 1, supporting hierarchical entities, zero-shot extraction and localizing the entities through the introduction of coordinate tokens.

VRD Representation Learning. As VRDs contain both textual and visual elements whose spatial position is crucial for their understanding, many works explore custom architectures and pretraining strategies to learn the relation between textual, layout and image modalities (Lee et al., 2023a; Appalaraju et al., 2023; Zhang et al., 2022). Xu

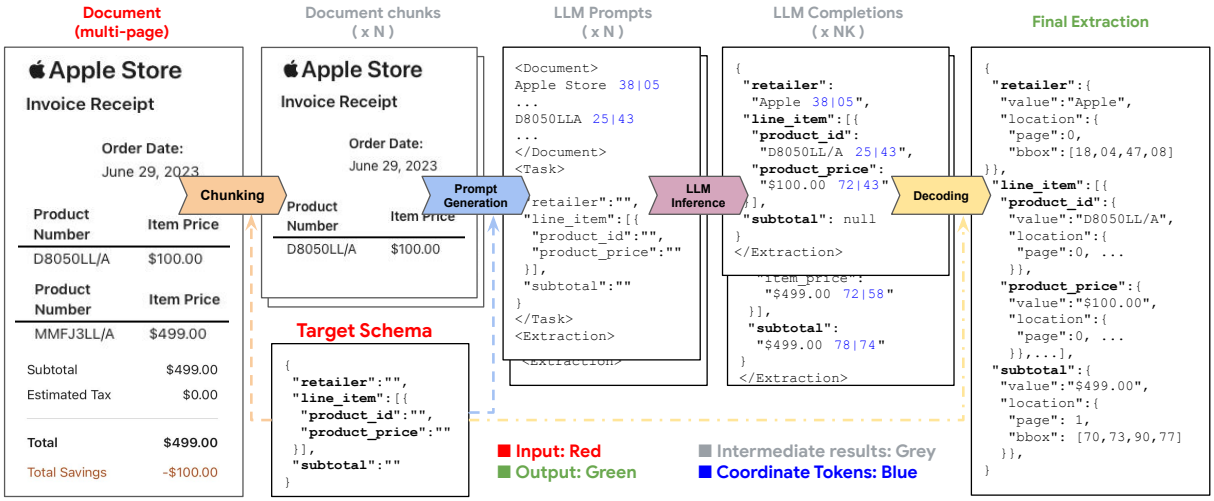


Figure 1: Overview of the LMDX methodology, decomposing the information extraction and localization task in 4 stages in order to frame it for an LLM. From the document, we generate LLM prompts containing both the text content and *coordinate tokens* (in color blue), which communicates the layout modality (needed for a high-quality extraction) and act as unique identifiers of the text segments. The prompts also contain the target schema, enabling zero-shot information extraction. The LLM completions, in JSON format, naturally support hierarchical entity extraction (e.g. *line_item*), and include both entity values and segment identifiers, enabling both entity localization (i.e. computing entity bounding box) and removing LLM hallucination through our decoding algorithm.

et al. (2020) uses a separate image encoder before adding the output as feature to the token encodings, while Huang et al. (2022) jointly models the page image patches alongside the tokens, using a word-patch alignment self-supervised pretraining task to learn the connection between the modalities. Hong et al. (2021) proposes to encode the relative 2D distances of text blocks in the attention of the transformer, and learning from unlabeled documents with an area-masking strategy. Kim et al. (2022); Lee et al. (2023b) foregoes the text modality completely, using a Vision Transformer encoder with an auto-regressive decoder pretrained on a pseudo-OCR and region masking task on large document image corpora. Unlike prior work, LMDX encodes the layout modality solely through text coordinate tokens, hence allows reusing LLMs with no architecture change and foregoing expensive vision encoders, while achieving state-of-the-art results.

LLMs for Extraction has mostly been studied in the text domain (Keraghel et al., 2024), either generally (Laskar et al., 2023) or domain-specific (De Toni et al., 2022; Hu et al., 2024). Wang et al. (2023b) uses a LLM to insert special tokens to mark the boundaries of target entities. Ashok and Lipton (2023) proposes a NER framework with in-context learning demonstrations, prompting the LLM to output an entities list with explanations justifying its matches with the provided entity definitions. Yet, LLMs remain underexplored for IE on VRDs. Wang et al. (2023a) uses unlabeled document corpora and turns existing labeled VRD understanding datasets in instruction tuning format, building a layout-aware LLM with various document understanding capabilities. In contrast, LMDX focuses on IE specifically, with an emphasis on hierarchical entity and entity localization support.

3 LMDX Methodology

Overall, our pipeline is divided into four stages: chunking, prompt generation, LLM inference and decoding, detailed in the following sections. An overview with a simple example can be found in Figure 1, with the input and output of each stage showcased. In this example, the target extraction schema contains two leaf entity types *retailer* and *subtotal*, and one hierarchical entity type *line_item*, composed of a *product_id* and a *product_price*.

Input Document. The input to our pipeline is the document's text segments (lines and words) along with their corresponding spatial position (bounding box) on the pages, typically obtained with an OCR service or a PDF rendering engine.

3.1 First Stage: Chunking

While some LLMs support long context (hundreds of thousands of tokens), not all LLMs can fit the entire document within its prompt, as documents can be hundreds of pages long. Thus, the document is divided into document chunks so that each is small

enough to be processed by the LLM. To achieve this, we first divide the document into individual pages, then we iteratively remove the last line segments until the prompt containing this chunk is below the maximum input token length of the LLM. Lastly, we group those removed lines as a new document page, and repeat the same logic until all chunks are below the input token limit of the LLM. At the end of this stage, we have N chunks. The decision to first divide the document by page stems from the observation that entities rarely cross page boundaries, and as such this chunking scheme will have minimal impact on the final extraction quality. The algorithm is described in pseudo-code in Appendix A.1.

3.2 Second Stage: Prompt Generation

The prompt generation stage takes in the N document chunks and creates a LLM prompt for each of them. As seen in Figure 2, our prompt design contains the document representation, a description of the task, and the target schema representation containing the entities to extract. XML-like tags are used to define the start and end of each component.

```
<Document>
{DOCUMENT_REPRESENTATION}
</Document>
<Task>
{TASK_DESCRIPTION}
{SCHEMA_REPRESENTATION}
</Task>
<Extraction>
```

Figure 2: Structure of the LLM prompts.

Document Representation. The chunk content is represented in the prompt as the concatenation of all its segment texts (lines or words from OCR), suffixed with the coordinates of those segments (derived from the bounding boxes) in the following format: `<segment text> XX|YY`$_{\texttt{segment}}$. Coordinate tokens, `XX` and `YY`, are built by normalizing the segment's X and Y coordinates, and quantizing them in B buckets, assigning the index of that bucket as the token for that coordinate.

Encoding the coordinates as tokens within the prompt allows us to communicate the layout modality to the LLM, without any change to its architecture. There are many variations to that scheme: using line versus words as segment, the granularity of the quantization, and the number of coordinates to use per segment (e.g. $[x_{\text{center}}, y_{\text{center}}]$ versus $[x_{\text{min}}, y_{\text{min}}, x_{\text{max}}, y_{\text{max}}]$). Appendix A.4 shows how those variations affect the prompt token length. Experimentally, we've found using line-level segments with 2 coordinates $[x_{\text{center}}, y_{\text{center}}]$ and $B = 100$ quantization buckets worked best, as detailed in Appendix A.12. Hence, we've adopted that coordinate tokenization scheme in our experiments.

Task Description. The task description is simply a short explanation of the task to accomplish. In our experiments, we set it to the following: *From the document, extract the text values and tags of the following entities:*.

Schema Representation. The schema is represented as a structured JSON object, where the keys are the entity types to be extracted, and the values correspond to their occurrence (single or multiple) and sub-entities (for hierarchical entities). For instance, *{"foo": "", "bar": [{"baz": []}]}* means that the LLM should extract only a single entity of type *foo* and multiple hierarchical entities of type *bar*, that could each hold multiple entities of type *baz*.

After this step, we have N prompts, one for each document chunk. An example of a prompt on a document can be found in Appendix A.8, Figure 10.

3.3 Completion Targets

In this section, we describe the expected LLM completion format, which can be observed in *LLM Completions* section of Figure 1. Like the schema, the completion is a JSON structured object with the keys being the entity types, and values being the extracted information from the document chunk. JSON was chosen as a format for the completion and schema since it supports hierarchical objects (hence hierarchical entities), is very token-efficient, and emitting JSON is within mainstream LLMs' capabilities (Sengottuvelu, 2023; OpenAI, 2023b). Note that the keys in the completion have the same ordering, occurrence and class (hierarchical or leaf) as the entity types in the schema. The values of leaf entities must follow a specific format:

$$\texttt{<text on segment}_1\texttt{> XX|YY}_{\texttt{segment}_1}\texttt{\textbackslash n}$$
$$\texttt{<text on segment}_2\texttt{> XX|YY}_{\texttt{segment}_2}\texttt{\textbackslash n ...}$$

An entity can span multiple (potentially disjoint) text segments (lines or words). For each segment of the entity, the value contains the entity text on that segment, along with the coordinate tokens of that segment, which act as a *segment identifier*, uniquely identifying the segment, and allowing us to localize the entities and ground the model prediction (e.g. making sure the extracted value is not a hallucination), as will be detailed in Section 3.5. Fi-

nally, missing entity types are explicitly completed by the model with $null$ for singular types, and [] for repeated types. Samples of completions can be found in Appendix A.8, Figure 10.

3.4 Third Stage: LLM Inference

In this stage of the pipeline, we run inference on the LLM with the N prompts. For each prompt, we sample K completions from the LLM (for a total of NK completions for the entire document) using Top_K sampling. This randomness in the sampling allows to do error correction (e.g. if a response is not valid JSON, have hallucinated segment coordinate identifier, etc), and increase the extraction quality as will be shown in Section 4.3. We use a fixed random seed to get a deterministic inference.

3.5 Fourth Stage: Decoding

In this stage (*Decoding* in Figure 1), we parse the LLM completions into entities and their locations.

Conversion to structured entities. We begin by parsing each LLM completion as a JSON object. Completions that fail to parse are discarded. For each key-value pair in the JSON object, we interpret the key as the entity type and parse the value to get the entity text and bounding box (as detailed in the next paragraph). Predicted entity types that are not in the target extraction schema are discarded. If the model unexpectedly predicts multiple values for single-occurrence entity types, we use the most frequent value as the final predicted value. Hierarchical JSON objects are recursively parsed as hierarchical entities in a similar manner. This algorithm is described in pseudo-code in Appendix A.3.

Entity Value Parsing. We expect the JSON value to include both text extractions and segment identifiers for each predicted entity, as described in Section 3.3. We first parse the value into its (*segment text*, *segment identifier*) pairs. For each pair, we look up the corresponding segment in the original document using the segment identifier and verify that the extracted text is *exactly* included on that segment. The entity is discarded if that verification fails, ensuring LMDX discards all LLM hallucinations. Finally, once we have the entity location on all its segments, we get the entity bounding box by computing the smallest bounding box encompassing all the words included in the entity. Entity values with any segments that fail to ground (invalid entity value format, non-existent segment identifier, or non-matching segment text) in the original document are discarded. The entity value parsing algorithm is described in pseudo-code in Appendix A.2, and parsing errors rates are detailed in Appendix A.10.

Prediction Merging. We first merge the predicted entities for the same document chunk from the K LLM completions through majority voting (Wang et al., 2022). For each entity type, we gather the predicted entities, including empty predictions, across the K completions. The most common prediction(s) are selected as the predicted value for that entity type. We then merge the predictions among the N document chunks by concatenating them to obtain the document level predictions.

Prediction Merging for hierarchical entities. For hierarchical entities, we use the entire predicted tree value from a single LLM completion, as this method best preserves the parent-child relationship predicted by the model. For each top-level hierarchical entity type, we perform majority voting on all affiliated leaf, intermediate and top-level entity types among K completions as if they are flattened. We then equally tally the votes to determine which completion to use for the prediction, and select the most common one for that hierarchical entity.

4 Evaluation

We seek to evaluate the effectiveness of LMDX on public IE benchmarks, and apply it to two distinct LLMs to validate the generality of the methodology: PaLM 2-S (Google et al., 2023) and Gemini Pro (Anil et al., 2023), which we call $\text{LMDX}_{\text{PaLM 2-S}}$ and $\text{LMDX}_{\text{Gemini Pro}}$ respectively.

Firstly, starting from their original checkpoint, we finetune those LLMs on the prompts and completions detailed in Section 3.2 and 3.3 on a data mixture containing a variety of *(document, schema, extraction)* tuples. In particular, this data mixture contains the *Payment* dataset (Majumder et al., 2020), along with a diverse set of publicly available PDF form templates obtained from government websites that we filled with synthetic data using an internal tool, and annotated for schema and entities to extract. The goal of this tuning is to obtain a *Base Entity Extractor* checkpoint by training the model to learn the IE task along with our desired extraction syntax. No document or schema contained in the base extraction training phase overlap with the documents and schemas used in our target benchmarks, hence we use those LLMs for zero-shot information extraction evaluation on the target benchmarks.

Finetuned Performance. We are also interested in evaluating how data-efficient LMDX is (e.g. how quickly it learns information extraction on a new document type). To answer this, starting from the *Base Entity Extractor* checkpoint, we finetune the LLM directly on the target benchmark.

Parameters. For training, we finetune using a batch size of 8, a dropout probability of 0.1 and a learning rate of 10^{-6} with a standard cross-entropy loss for 4000 steps on TPUv4 (Jouppi et al., 2023). Once training is done, for finetuned experiments, we select the checkpoint with the lowest loss on the dev set, and report performance on the test set. For LLM inference, we use a temperature of 0.5 and a Top_K of 40, sampling 16 responses for each chunk processed by the LLM, as described in Section 3.4. Finally, for both training and inference, we use an input token length of 6144 and output token length of 2048. We use line-level segments and only two coordinates $[x_{\text{center}}, y_{\text{center}}]$ with 100 quantization buckets, as supported by Appendix A.12.

4.1 Benchmarks

Visually Rich Document Understanding (VRDU). Wang et al. (2023d) introduces two public visually-rich documents IE benchmarks: *Registration Form*, containing 6 semantically rich entity types, and *Ad-buy Form*, containing 14 entity types with one hierarchical *line_item* entity type. For each benchmark, VRDU proposes samples of 10, 50, 100 and 200 train documents with high-quality OCR[1] which we use to evaluate the data efficiency of LMDX. It also offers different tasks to evaluate the generalization powers of extraction systems: Single Template Learning (STL) where train/test splits share the same single template, Unseen Template Learning (UTL) where train/test contain disjoint sets of templates, and Mixed Template Learning (MTL) where train/test contain overlapping sets of templates. We report Micro-F1 through the provided evaluation tool. For VRDU, we only run the finetuning experiments using $\text{LMDX}_{\text{PaLM 2-S}}$, given the significant cost of finetuning on all its tasks and train split sizes.

Consolidated Receipt Dataset (CORD).[2] Park et al. (2019) introduces a benchmark of Indonesian receipts from shops and restaurants, with a target schema of 30 fine-grained entities, grouped into *menu*, *total* and *subtotal* hierarchical entities. We adopt the evaluation tool from prior work (Kim et al., 2022) and report Micro-F1 on that benchmark. For our experiments, we use the official $train$ ($|\mathcal{D}| = 800$), dev and $test$ splits. To evaluate LMDX's data efficiency, we further sample the first $|\mathcal{D}| = 10/50/100/200$ documents from the $train$ split. For each of those data setup, we finetune LMDX for 12000 steps. For comparison, we also train and evaluate state-of-the-art baselines $\text{LayoutLMv3}_{\text{LARGE}}$ and Donut. Those baselines are detailed in Appendix A.7.

Baselines In the zero-shot setting, we compare LMDX to other LLM baselines: GPT-3.5, Gemini Pro (Anil et al., 2023), and PaLM 2-S (Google et al., 2023) that we prompt with the raw OCR text and IE instruction (called **GPT-3.5$_{\text{+OCR}}$**, **Gemini Pro$_{\text{+OCR}}$** and **PaLM 2-S$_{\text{+OCR}}$** respectively). We also compare LMDX with strong Vision-Language models: LLaVA-v1.5-13B (Liu et al., 2023), Gemini Pro, and GPT-4V (OpenAI, 2023a) that we prompt with the document page image and IE instructions (called **LLaVA-v1.5-13B$_{\text{+Image}}$**, **GPT-4V$_{\text{+Image}}$**, and **Gemini Pro$_{\text{+Image}}$**). Those baselines are fully detailed in Appendix A.6. Unlike LMDX, those large model baselines do not localize their predictions. In the finetuned setting, we compare $\text{LMDX}_{\text{PaLM 2-S}}$ to popular VRD IE baselines. For VRDU, we compare to its published baselines (Wang et al., 2023d), $\text{LayoutLM}/\text{v2}/\text{v3}$ and FormNet. For CORD, we train and evaluate state-of-the-art baselines LayoutLMv3 and Donut. Those baselines are detailed in Appendix A.7. Following prior work (Lee et al., 2022, 2023a; Xu et al., 2020, 2021), for all models leveraging the text modality, we use the benchmarks' provided OCR, ensuring a fair comparison.

4.2 Results

VRDU results are presented in Table 2. In zero-shot ($|\mathcal{D}| = 0$) setting, $\text{LMDX}_{\text{PaLM 2-S}}$ and $\text{LMDX}_{\text{Gemini Pro}}$ have higher extraction quality than all other large models baselines, including the ones using the same LLM and OCR or image, showing improvements brought by the LMDX methodology itself. In finetuned setting on VRDU, $\text{LMDX}_{\text{PaLM 2-S}}$ is much more data efficient than the baselines: it is at 5.06% Micro-F1 of its peak performance at 10 training documents for Registration Form Mixed Template (87.72% vs 92.78% Micro-F1) while LayoutLMv2, the strongest finetuned baseline, is within 19.75% of its peak performance (69.44% vs 89.19% Micro-F1), showcasing that it learns extraction on a new document type much

[1] https://cloud.google.com/vision/docs/ocr

[2] https://huggingface.co/datasets/naver-clova-ix/cord-v1

Table 2: Results of $\text{LMDX}_{\text{PaLM 2-S}}$ and $\text{LMDX}_{\text{Gemini Pro}}$ on the different tasks and training data size setups $|\mathcal{D}|$ of VRDU, with best and second best performing model results in bold and underlined respectively, with Micro-F1 reported. We specify the modalities leveraged by each model ($T \rightarrow Text$, $L \rightarrow Layout$, $I \rightarrow Image$) and whether their entities are localized.

$\lvert\mathcal{D}\rvert$	Model	Modality	Localized?	Registration Form			Ad-buy Form		
				Single	Unseen	Mixed Template	Unseen	Mixed Template	
				Micro-F1	Micro-F1	Micro-F1	Micro-F1	Micro-F1	Line Item F1
0	LLaVA-v1.5-13B$_{\text{+Image}}$	*I*	✗	5.29	5.05	5.00	0.38	0.34	0.00
	GPT-4V$_{\text{+Image}}$	*I*	✗	68.97	69.44	65.34	31.84	31.95	4.45
	Gemini Pro$_{\text{+Image}}$	*I*	✗	53.90	53.72	48.60	15.24	15.38	0.91
	Gemini Pro$_{\text{+OCR}}$	*T*	✗	73.62	73.66	69.41	32.90	34.46	19.25
	PaLM 2-S$_{\text{+OCR}}$	*T*	✗	62.80	63.51	59.78	29.70	30.24	9.86
	GPT-3.5$_{\text{+OCR}}$	*T*	✗	67.23	67.49	63.86	29.84	30.05	7.65
	LMDX$_{\text{PaLM 2-S}}$	*T+L*	✓	<u>73.81</u>	<u>74.94</u>	<u>71.65</u>	**39.33**	**39.74**	<u>21.21</u>
	LMDX$_{\text{Gemini Pro}}$	*T+L*	✓	**76.78**	**77.18**	**75.15**	<u>37.94</u>	<u>38.02</u>	**23.29**
10	FormNet	*T+L*	✓	74.22	50.53	63.61	20.28	20.47	5.72
	LayoutLM	*T+L+I*	✓	65.91	25.54	36.41	19.92	20.20	6.95
	LayoutLMv2	*T+L+I*	✓	<u>80.05</u>	<u>54.21</u>	<u>69.44</u>	<u>25.17</u>	<u>25.36</u>	<u>9.96</u>
	LayoutLMv3	*T+L+I*	✓	72.51	21.17	60.72	10.01	10.16	5.92
	LMDX$_{\text{PaLM 2-S}}$	*T+L*	✓	**90.88**	**86.87**	**87.72**	**54.82**	**54.35**	**39.35**
50	FormNet	*T+L*	✓	<u>89.38</u>	<u>68.29</u>	<u>85.38</u>	39.52	40.68	19.06
	LayoutLM	*T+L+I*	✓	86.21	55.86	80.15	38.42	39.76	19.50
	LayoutLMv2	*T+L+I*	✓	88.68	61.36	84.13	<u>41.59</u>	<u>42.23</u>	<u>20.98</u>
	LayoutLMv3	*T+L+I*	✓	87.24	47.85	81.36	38.43	39.49	19.53
	LMDX$_{\text{PaLM 2-S}}$	*T+L*	✓	**93.06**	**88.43**	**91.42**	**75.70**	**75.08**	**65.42**
100	FormNet	*T+L*	✓	<u>90.91</u>	<u>72.58</u>	88.13	39.88	40.38	18.80
	LayoutLM	*T+L+I*	✓	88.70	63.68	86.02	41.46	42.38	21.26
	LayoutLMv2	*T+L+I*	✓	90.45	65.96	<u>88.36</u>	<u>44.35</u>	<u>44.97</u>	<u>23.52</u>
	LayoutLMv3	*T+L+I*	✓	89.23	57.69	87.32	41.54	42.63	22.08
	LMDX$_{\text{PaLM 2-S}}$	*T+L*	✓	**93.97**	**89.70**	**92.41**	**75.99**	**78.05**	**69.77**
200	FormNet	*T+L*	✓	<u>92.12</u>	<u>77.29</u>	<u>90.51</u>	42.87	43.23	21.86
	LayoutLM	*T+L+I*	✓	90.47	70.47	87.94	44.18	44.66	23.90
	LayoutLMv2	*T+L+I*	✓	91.41	72.03	89.19	<u>46.31</u>	<u>46.54</u>	<u>25.46</u>
	LayoutLMv3	*T+L+I*	✓	90.89	62.58	89.77	44.43	45.16	24.51
	LMDX$_{\text{PaLM 2-S}}$	*T+L*	✓	**93.97**	**90.22**	**92.78**	**78.42**	**79.82**	**72.09**

faster. Moreover, $\text{LMDX}_{\text{PaLM 2-S}}$ generalizes better to unseen templates than finetuned baselines: on Registration Form, $\text{LMDX}_{\text{PaLM 2-S}}$ has a drop less than 5% Micro-F1 on Unseen Template compared to Single Template across data regimes, while LayoutLMv2 see a drop between 19% and 27%.

On CORD (in Table 3), we observe similar trends, reaching state-of-the-art on all but one data regime, highlighting the generality of the results.

Performance on Hierarchical Entities. To showcase extraction quality on hierarchical entities, we display in Table 2 the F1 score on Ad-buy Form Mixed's *line_item* entity type. Overall, LMDX has much higher Line Item F1 than the baselines for all data regimes. In particular, $\text{LMDX}_{\text{PaLM 2-S}}$ has similar Line Item F1 at zero-shot than the best finetuned baseline at 200 train documents (21.21%

Table 3: Results of $\text{LMDX}_{\text{PaLM 2-S}}$ and $\text{LMDX}_{\text{Gemini Pro}}$ on the different training data size setups $|\mathcal{D}|$ of CORD, compared to zero-shot (Large Models) and finetuned (LayoutLMv3 and Donut) baselines. Micro-F1 is reported, with best and second best performing model results in bold and underlined respectively.

Model	Modality	Localized?	$\lvert D\rvert$=0	$\lvert D\rvert$=10	$\lvert D\rvert$=50	$\lvert D\rvert$=100	$\lvert D\rvert$=200	$\lvert D\rvert$=800
LLaVA-v1.5-13B	*I*	✗	5.97	-	-	-	-	-
GPT-4V$_{\text{+Image}}$	*I*	✗	64.05	-	-	-	-	-
Gemini Pro$_{\text{+Image}}$	*I*	✗	47.12	-	-	-	-	-
Gemini Pro$_{\text{+OCR}}$	*T*	✗	59.57	-	-	-	-	-
PaLM 2-S$_{\text{+OCR}}$	*T*	✗	55.85	-	-	-	-	-
GPT-3.5$_{\text{+OCR}}$	*T*	✗	48.92	-	-	-	-	-
Donut	*I*	✗	0.00	26.15	65.68	71.81	75.85	81.55
LayoutLMv3	*T+L+I*	✓	0.00	74.04	85.78	90.39	<u>93.59</u>	**95.66**
LMDX$_{\text{PaLM 2-S}}$	*T+L*	✓	**66.95**	**90.02**	<u>91.40</u>	**91.48**	93.40	94.51
LMDX$_{\text{Gemini Pro}}$	*T+L*	✓	<u>66.03</u>	<u>89.45</u>	**91.66**	<u>91.16</u>	**93.76**	<u>95.57</u>

Table 4: Entity Localization Accuracy on Registration Form and Ad-Buy Form Mixed Benchmarks, for models supporting localization. Best result is in bold.

	Model	$\|\mathcal{D}\|$=0	$\|\mathcal{D}\|$=10	$\|\mathcal{D}\|$=50	$\|\mathcal{D}\|$=100	$\|\mathcal{D}\|$=200
Reg. Form	LayoutLM	N/A	98.71	99.69	99.63	99.69
	LayoutLMv2	N/A	99.00	99.54	99.72	99.75
	LayoutLMv3	N/A	99.20	99.39	99.72	99.67
	LMDX$_{\text{PaLM 2-S}}$	93.21	**99.75**	**99.87**	**99.92**	**99.87**
	LMDX$_{\text{Gemini Pro}}$	**94.43**	-	-	-	-
Ad-buy Form	LayoutLM	N/A	92.60	95.24	95.09	95.38
	LayoutLMv2	N/A	93.95	95.64	95.72	95.78
	LayoutLMv3	N/A	90.68	95.28	95.88	95.95
	LMDX$_{\text{PaLM 2-S}}$	88.18	**94.51**	**98.28**	**98.69**	**98.65**
	LMDX$_{\text{Gemini Pro}}$	**92.51**	-	-	-	-

versus 25.46% respectively). With all the training data, LMDX$_{\text{PaLM 2-S}}$ scores a 72.09% Line Item F1, an absolute improvement of 46.63% over the best baseline LayoutLMv2. Finally, as LMDX encodes the layout modality, it possesses much higher zero-shot Line Item F1 than large models baselines.

Entity Localization Accuracy. In order to evaluate the localization quality independently of the extraction quality, we compute the Localization Accuracy of LMDX and all baselines that can localize entities using the formula: $Accuracy_{Localization} = \frac{N_{E+L}}{N_E}$ where N_{E+L} is the number of entities correctly extracted and localized, and N_E is the number of entities correctly extracted. Since LMDX localizes at the line level, localization verification is done at the line-level as well, i.e. localization is considered correct if the prediction bounding box is covered by the groundtruth line-level bounding box by more than 80%. We present the results in Table 4. Overall, LMDX$_{\text{PaLM 2-S}}$ and LMDX$_{\text{Gemini Pro}}$ can localize their predictions reliably at the line-level with the segment identifiers, with 88%-94% accuracy at zero-shot, and 98%-99% in finetuned cases, which is slightly higher than baselines.

4.3 Ablation Study

In this section, we ablate different facets of the LMDX methodology to highlight their importance. The results can be found in Table 5. For all ablations, we evaluate LMDX$_{\text{PaLM 2-S}}$ on the VRDU Ad-Buy Form Mixed Template task at $|\mathcal{D}| = 10$ data size, only changing the ablated facet.

Effects of Base Entity Extraction Training. In this ablation, we remove the initial training on the varied data mixture and directly finetune on the VRDU target task. As seen in Table 5, skipping that training leads to -11.44% micro-F1 as the model has to learn from scratch the task, the desired completion format and coordinate tokens' semantics.

Effects of Coordinate Tokens. In this ablation, we replace the coordinate tokens, which communicate the position of each line within the document, by the index of that line. This index still acts as a unique identifier for the line segment (required for entity localization) but does not communicate any position information. An example of a prompt with line index can be found in Appendix A.8 Figure 11, and per-entity F1 can be found in Appendix A.14 Table 9. As seen in Table 5, the coordinate tokens are crucial to quality, leading +14.98% micro-F1.

Effects of Sampling Strategy. In this ablation, we discard our strategy of sampling $K = 16$ completions per chunk, and instead sample a single response. As seen in Table 5, this leads to a 1.5% drop in micro-F1. While overall minor for quality, the sampling strategy corrects extraction format mistakes (see parsing error rates in Appendix A.10), leading to a successful extraction on all documents.

Effects of Missing Entity Types. In this ablation, we study the effect of having the model's completions skip missing entity types in the completions instead of explicitly outputting "$type$" : $null$ for those (See example in Appendix A.8, Figure 12). As seen in Table 5, this leads to a 6.77% drop in micro-F1 over explicitly outputting missing types. We hypothesize that this is due to the fact that having completions skip missing types means that, during response generation, the model has to choose with a single token computation budget within the N remaining entity types which one is the next present (essentially a N-way classification). Explicitly emitting missing entity types means the model only has to copy the types directly from the schema in the prompt, and has to do 2-way classification within a single token computation budget to declare if an entity is present or not (e.g. emit token $null$ if entity is missing or " if present), which is an easier task.

Table 5: Ablations of LMDX's core designs. Ablations are done on VRDU Ad-Buy Mixed Template with LMDX$_{\text{PaLM 2-S}}$ with at $|\mathcal{D}| = 10$ data size. All components contribute to the final performance.

LMDX Micro-F1 (Δ)	**Without EE Training**	**Without Coordinate Tokens**	**Without Sample Strategy**	**Without Missing Types**
54.35	42.91 (-11.44)	39.37 (-14.98)	52.85 (-1.50)	47.58 (-6.77)

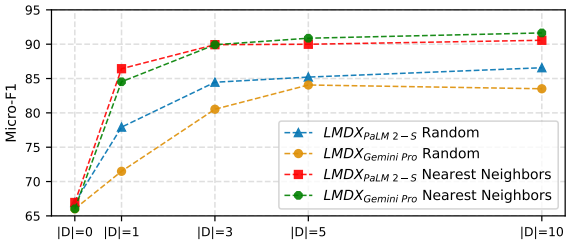


Figure 3: In-Context Learning results on CORD with random and nearest neighbors retrieval methods for $\text{LMDX}_{\text{PaLM 2-S}}$ and $\text{LMDX}_{\text{Gemini Pro}}$.

4.4 In-context Learning Performance

In this section, we study how in-context learning (ICL) compares to finetuning. To do so, we test two methodologies: *Random*, which randomly selects $|\mathcal{D}|$ documents and extractions from the train set, and *Nearest Neighbors*, which uses similarity based on SentenceT5 embeddings (Ni et al., 2021) to retrieve $|\mathcal{D}|$ documents to add in the LLM context. The results on CORD for $\text{LMDX}_{\text{PaLM 2-S}}$ and $\text{LMDX}_{\text{Gemini Pro}}$ are shown in Figure 3. Overall, while both methods increase the performance significantly, nearest neighbors shows a clear advantage, matching the best random ICL performance with only a single in-context example (86.43% versus 86.57% micro-F1 for $\text{LMDX}_{\text{PaLM 2-S}}$), and matching the finetuned performance at $|\mathcal{D}| = 10$ examples (90.33% versus 90.02% micro-F1), as examples from the same template are retrieved (see Appendix A.9). Beyond $|\mathcal{D}| = 10$, the quality plateaus as no more example fit in the prompt.

5 Conclusion

In this paper, we have introduced LMDX, a methodology that enables using LLMs for information extraction on visually rich documents. With its coordinate tokens and decoding strategy, LMDX allows the high-quality extraction of singular, repeated and hierarchical entities, while localizing the entities in the document. LMDX is data efficient, and even allows extraction at zero-shot on entirely new document types and schemas. LMDX can benefit from orthogonal research fields, and we continue the discussion in the Limitations section.

Acknowledgements

The authors would like to thank Tania Bedrax-Weiss, Riham Mansour, Slav Petrov, Yunhsuan Sung, Mike Kwong and Chun-Liang Li for their valuable feedback on the experiments and paper. The authors also thank Nikolai Glushnev for discussions, along with the help naming LMDX.

Limitations

We acknowledge the limitations of LMDX from the following aspects to inspire future research in the field of information extraction and localization.

Firstly, LMDX's input is text lines and their bounding boxes, usually coming from OCR. This means that LMDX can not extract non-textual entities thus would not be able to extract an entity that would be an image embedded in a document (e.g. a product_image entity in a product webpage). This also limits performance in high-data scenarios, as all page image information is discarded. Furthermore, such input means that LMDX is sensitive to errors from the OCR process (wrong reading order, incorrect line grouping, undetected text and erroneously recognized characters). Qualitatively, we have found that a common error type for LMDX is caused by OCR grouping multiple semantically different segments together (we give a deeper analysis with concrete examples in Appendix A.11). Thus, techniques aiming to improve this error type would be a worthwhile future research direction.

Moreover, LMDX's localization mechanism is applied at the line level, where we verify that the predicted text is indeed present on the line. If the entity text appears multiple times on the line, we don't have a definitive way to choose the correct text. Thus, LMDX's localization and bounding boxes are not reliable beyond line-level granularity. While sufficient for greatly speeding up human-in-the-loop interactions like prediction auditing/review, getting entity bounding boxes precise at character-level would be beneficial, more natural-looking, and a worthy research direction.

Lastly, LMDX relies on LLMs supporting thousands of tokens in input and output (as detailed in Appendix A.4), which is both computationally expensive and slow, requiring the use of hardware acceleration for acceptable latency and throughput. We showcase a latency comparison between popular solutions in Appendix A.13. General research in accelerating LLM inference (Shazeer, 2019; Ainslie et al., 2023; Leviathan et al., 2023; Hong et al., 2023) would make LMDX more cost-effective in production setting. Specifically for LMDX, the coordinate tokens represent a large part of the total number of tokens, so research on minimizing their number (e.g. by introducing dedicated coordinate tokens within the LLM vocabulary) would yield significant savings and be a worthwhile direction for future work.

References

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*.

Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Slav Petrov, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy P. Lillicrap, Angeliki Lazaridou, Orhan Firat, James Molloy, Michael Isard, Paul Ronald Barham, Tom Hennigan, Benjamin Lee, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, Ryan Doherty, Eli Collins, Clemens Meyer, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha Goel, George Tucker, Enrique Piqueras, Maxim Krikun, Iain Barr, Nikolay Savinov, Ivo Danihelka, Becca Roelofs, Anaïs White, Anders Andreassen, Tamara von Glehn, Lakshman Yagati, Mehran Kazemi, Lucas Gonzalez, Misha Khalman, Jakub Sygnowski, and et al. 2023. Gemini: A family of highly capable multimodal models. *CoRR*, abs/2312.11805.

Srikar Appalaraju, Bhavan Jasani, Bhargava Urala Kota, Yusheng Xie, and R. Manmatha. 2021. Docformer: End-to-end transformer for document understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 993–1003.

Srikar Appalaraju, Peng Tang, Qi Dong, Nishant Sankaran, Yichu Zhou, and R. Manmatha. 2023. Docformerv2: Local features for document understanding.

Dhananjay Ashok and Zachary C. Lipton. 2023. Promptner: Prompting for named entity recognition.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Francesco De Toni, Christopher Akiki, Javier De La Rosa, Clémentine Fourrier, Enrique Manjavacas, Stefan Schweter, and Daniel Van Strien. 2022. Entities, dates, and languages: Zero-shot on historical texts with t0. In *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*, pages 75–83, virtual+Dublin. Association for Computational Linguistics.

Rohan Anil Google, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernandez Abrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vlad Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, Guy Gur-Ari, Steven Hand, Hadi Hashemi, Le Hou, Joshua Howland, Andrea Hu, Jeffrey Hui, Jeremy Hurwitz, Michael Isard, Abe Ittycheriah, Matthew Jagielski, Wenhao Jia, Kathleen Kenealy, Maxim Krikun, Sneha Kudugunta, Chang Lan, Katherine Lee, Benjamin Lee, Eric Li, Music Li, Wei Li, YaGuang Li, Jian Li, Hyeontaek Lim, Hanzhao Lin, Zhongtao Liu, Frederick Liu, Marcello Maggioni, Aroma Mahendru, Joshua Maynez, Vedant Misra, Maysam Moussalem, Zachary Nado, John Nham, Eric Ni, Andrew Nystrom, Alicia Parrish, Marie Pellat, Martin Polacek, Alex Polozov, Reiner Pope, Siyuan Qiao, Emily Reif, Bryan Richter, Parker Riley, Alex Castro Ros, Aurko Roy, Brennan Saeta, Rajkumar Samuel, Renee Shelby, Ambrose Slone, Daniel Smilkov, David R. So, Daniel Sohn, Simon Tokumine, Dasha Valter, Vijay Vasudevan, Kiran Vodrahalli, Xuezhi Wang, Pidong Wang, Zirui Wang, Tao Wang, John Wieting, Yuhuai Wu, Kelvin Xu, Yunhan Xu, Linting Xue, Pengcheng Yin, Jiahui Yu, Qiao Zhang, Steven Zheng, Ce Zheng, Weikang Zhou, Denny Zhou, Slav Petrov, and Yonghui Wu. 2023. Palm 2 technical report.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training compute-optimal large language models.

Ke Hong, Guohao Dai, Jiaming Xu, Qiuli Mao, Xiuhong Li, Jun Liu, Kangdi Chen, Hanyu Dong, and Yu Wang. 2023. Flashdecoding++: Faster large language model inference on gpus. *arXiv preprint arXiv:2311.01282*.

Teakgyu Hong, Donghyun Kim, Mingi Ji, Wonseok Hwang, Daehyun Nam, and Sungrae Park. 2021. BROS: A layout-aware pre-trained language model for understanding documents. *CoRR*, abs/2108.04539.

Yan Hu, Qingyu Chen, Jingcheng Du, Xueqing Peng,

Vipina Kuttichi Keloth, Xu Zuo, Yujia Zhou, Zehan Li, Xiaoqian Jiang, Zhiyong Lu, Kirk Roberts, and Hua Xu. 2024. Improving large language models for clinical named entity recognition via prompt engineering.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions.

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

Norm Jouppi, George Kurian, Sheng Li, Peter Ma, Rahul Nagarajan, Lifeng Nai, Nishant Patil, Suvinay Subramanian, Andy Swing, Brian Towles, et al. 2023. Tpu v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, pages 1–14.

Imed Keraghel, Stanislas Morbieu, and Mohamed Nadif. 2024. A survey on recent advances in named entity recognition. *arXiv preprint arXiv:2401.10825*.

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 (ECCV)*.

Md Tahmid Rahman Laskar, M Saiful Bari, Mizanur Rahman, Md Amran Hossen Bhuiyan, Shafiq Joty, and Jimmy Xiangji Huang. 2023. A systematic study and comprehensive evaluation of chatgpt on benchmark datasets.

Chen-Yu Lee, Chun-Liang Li, Timothy Dozat, Vincent Perot, Guolong Su, Nan Hua, Joshua Ainslie, Renshen Wang, Yasuhisa Fujii, and Tomas Pfister. 2022. FormNet: Structural encoding beyond sequential modeling in form document information extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3735–3754, Dublin, Ireland. Association for Computational Linguistics.

Chen-Yu Lee, Chun-Liang Li, Chu Wang, Renshen Wang, Yasuhisa Fujii, Siyang Qin, Ashok Popat, and Tomas Pfister. 2021. Rope: reading order equivariant positional encoding for graph-based document information extraction. In *ACL*.

Chen-Yu Lee, Chun-Liang Li, Hao Zhang, Timothy Dozat, Vincent Perot, Guolong Su, Xiang Zhang, Kihyuk Sohn, Nikolay Glushnev, Renshen Wang, Joshua Ainslie, Shangbang Long, Siyang Qin, Yasuhisa Fujii, Nan Hua, and Tomas Pfister. 2023a. FormNetV2: Multimodal graph contrastive learning for form document information extraction. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9011–9026, Toronto, Canada. Association for Computational Linguistics.

Kenton Lee, Mandar Joshi, Iulia Turc, Hexiang Hu, Fangyu Liu, Julian Eisenschlos, Urvashi Khandelwal, Peter Shaw, Ming-Wei Chang, and Kristina Toutanova. 2023b. Pix2struct: Screenshot parsing as pretraining for visual language understanding.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2023. Improved baselines with visual instruction tuning.

Bodhisattwa Prasad Majumder, Navneet Potti, Sandeep Tata, James Bradley Wendt, Qi Zhao, and Marc Najork. 2020. Representation learning for information extraction from form-like documents. In *ACL*.

Jianmo Ni, Gustavo Hernández Ábrego, Noah Constant, Ji Ma, Keith B Hall, Daniel Cer, and Yinfei Yang. 2021. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. *arXiv preprint arXiv:2108.08877*.

OpenAI. 2023a. Gpt-4 technical report.

OpenAI. 2023b. Json mode. `https://platform.openai.com/docs/guides/text-generation/json-mode`. Accessed: 2024-02-15.

Rasmus Berg Palm, Ole Winther, and Florian Laws. 2017. Cloudscan - a configuration-free invoice analysis system using recurrent neural networks. In *Proceedings of 2017 14th IAPR International Conference on Document Analysis and Recognition*, pages 406–413, United States. IEEE.

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*.

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. In *Document Analysis and Recognition – ICDAR 2021*, pages 732–747, Cham. Springer International Publishing.

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. *Journal of Machine Learning Research*, 21(140):1–67.

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In *Third Workshop on Very Large Corpora*.

Rahul Sengottuvelu. 2023. Jsonformer. https://github.com/1rgs/jsonformer. Accessed: 2024-02-15.

Noam Shazeer. 2019. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Dongsheng Wang, Natraj Raman, Mathieu Sibue, Zhiqiang Ma, Petr Babkin, Simerjot Kaur, Yulong Pei, Armineh Nourbakhsh, and Xiaomo Liu. 2023a. Docllm: A layout-aware generative language model for multimodal document understanding.

Shuhe Wang, Xiaofei Sun, Xiaoya Li, Rongbin Ouyang, Fei Wu, Tianwei Zhang, Jiwei Li, and Guoyin Wang. 2023b. Gpt-ner: Named entity recognition via large language models.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Zifeng Wang, Zizhao Zhang, Jacob Devlin, Chen-Yu Lee, Guolong Su, Hao Zhang, Jennifer Dy, Vincent Perot, and Tomas Pfister. 2023c. QueryForm: A simple zero-shot form entity query framework. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4146–4159, Toronto, Canada. Association for Computational Linguistics.

Zilong Wang, Yichao Zhou, Wei Wei, Chen-Yu Lee, and Sandeep Tata. 2023d. Vrdu: A benchmark for visually-rich document understanding. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, page 5184–5193, New York, NY, USA. Association for Computing Machinery.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are zero-shot learners.

Yang Xu, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, Wanxiang Che, Min Zhang, and Lidong Zhou. 2021. Layoutlmv2: Multi-modal pre-training for visually-rich document understanding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL) 2021*.

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

Zhenrong Zhang, Jiefeng Ma, Jun Du, Licheng Wang, and Jianshu Zhang. 2022. Multimodal pre-training based on graph attention network for document understanding.

A Appendix

A.1 Chunking algorithm

Algorithm 1 Document Chunking

function CHUNK(D, L, F) ▷ D is a document containing multiple pages. L is token limit.
▷ F is a function that outputs prompt token length given some segments (e.g. lines).
    $C = \phi$ ▷ C is to record all produced chunks.
    for $i = 1$ to $|D.pages|$ **do**
        $S = D.pages[i].segments$
        while $S \neq \phi$ **do**
            for $j = |S|$ to 1 **do** ▷ Start pruning from the end of the page.
                if $F(S[1:j]) \leq L$ **then**
                    $C = C \cup \{S[1:j]\}$
                    $S = S[j+1:|S|]$ ▷ Continue to work on the rest of the segments.
                    Exit for loop
                end if
            end for
        end while
    end for
    return C
end function

A.2 Entity value parsing algorithm

Algorithm 2 Entity Value Parsing

function PARSEENTITYVALUE(D, E) ▷ D is a document chunk.
▷ E is raw extraction results for one entity type parsed from one LLM sample.
    $G = \phi$ ▷ G is to record all parsed entity values.
    $R = Regex(\text{“}(\backslash d \backslash | \backslash d \backslash d)\text{”})$ ▷ R is a regex that captures the segment identifiers.
    $M = \{\text{“}s.x|s.y\text{”} \mapsto s | s \in D.segments\}$ ▷ M holds a mapping between segment id and segment.
    for i = 1 to $|E|$ **do**
        $W = \phi$ ▷ W is to hold all words for this entity.
        $P = R.split(E[i])$ ▷ P is expected to be interleaved text values and segment ids.
        for j = 1 to $|P|/2$ **do**
            if $P[j*2] \notin M$ **then**
                Go to next i ▷ Segment ID is hallucinated. Grounding failure.
            end if
            $S = M[P[j*2]]$ ▷ Retrieve the stored segment from M with parsed segment ID.
            $T = P[j*2-1]$ ▷ T is to hold the predicted text.
            if T not substring of S **then**
                Go to next i ▷ Grounding failure, skip the current entity.
            end if
            $W = W \cup (S \cap T)$
        end for
        $G'.value = \bigcup_{w \in W} w.text\_value$ ▷ G' is to hold the entity to return.
        $G'.bounding\_box = \{\min(b.x), \min(b.y), \max(b.x), \max(b.y)\}_{w \in W, b = w.bounding\_box}$
        $G = G \cup \{G'\}$
    end for
    return G
end function

A.3 Decoding algorithm

Algorithm 3 Responses Decoding

function DECODEFORTYPE(J, T, D) ▷ J is one or more JSON objects.
▷ T is an entity type.
▷ D is a document chunk.
  $E = \phi$ ▷ E is to record all parsed and grounded entities.
  for j = 1 to $|J|$ **do**
    $J' = J[j][T.type]$ ▷ J' is to hold entities for T's type before grounding.
    if $T.subtypes = \phi$ **then** ▷ T is leaf entity type.
      $E = E \cup ParseEntityValue(D, J')$
    else ▷ T is hierarchical entity type.
      $E'.subtypes = \bigcup_{T' \in T.subtypes} DecodeForType(J', T', D)$ ▷ E' is hierarchical entity.
      $E = E \cup \{E'\}$
    end if
  end for
  return E
end function

function MAJORITYVOTING(T, E) ▷ T is an entity type.
▷ E is a 2D vector of entities of type T from all LLM responses.
  $V = [0, 0, ..., 0] \in \mathbb{R}^{|E|}$ ▷ V is to record all votes.
  $L = \{T\}$
  while $L \neq \phi$ **do**
    $T' = L[0]$
    $E' = \phi$
    for j = 1 to $|E|$ **do**
      $E' = E' \cup \{e | e \in E[j], e.type = T'\}$ ▷ $E'[j]$ holds entities with type T' from E[j].
    end for
    for i = 1 to $|E'|$ - 1 **do**
      for $j = i + 1$ to $|E'|$ **do**
        if $E'[i] = E'[j]$ **then**
          $V[i] = V[i] + 1$
          $V[j] = V[j] + 1$
        end if
      end for
    end for
    $L = L[1 : |L|]$ ▷ Remove T' and inject its sub-types for recursion.
    $L = L \cup T'.subtypes$
  end while
  return $E[argmax(V)]$ ▷ Return the entity values with the highest votes.
end function

function DECODEALLSAMPLES(S, T, D) ▷ S is all LLM response samples on D.
▷ T is a list of entity types.
▷ D is a document chunk.
  return $\bigcup_{T' \in T} MajorityVoting(\bigcup_{S' \in S} DecodeForType(ParseJson(S'), T', D))$
end function

A.4 Token Length Statistics

Table 6 details the token length (50^{th} and 99^{th} percentiles) of the prompt and completion targets for $\text{LMDX}_{\text{PaLM 2-S}}$ for the train split of datasets used in our experiments. We select the line level segment, 2 coordinate scheme, no JSON indentation so that all datasets fit within our 6144 prompt token length and 2048 output token length.

A.5 Schemas

In this section, we present the schemas used for the experiments of this paper. The schema for VRDU Ad-Buy Form, VRDU Registration Form, and CORD can be found in Figure 4, Figure 5 and Figure 6 respectively.

Table 6: Prompt and target token length of different coordinate-as-tokens schemes on VRDU and CORD benchmarks, using the vocabulary of PaLM 2-S. We vary the number of coordinates and their quantization buckets in the localization tags, the segment level (e.g. line versus word), chunking style (e.g. page versus max input tokens) and JSON indentation in the schema and completion targets.

VRDU Ad-Buy Form

# Coord.	# Quant.	Segment	Chunking	JSON Indent	Input		Target	
					50^{th}	99^{th}	50^{th}	99^{th}
2	100	Line	Page	None	2377	3920	602	1916
2	100	Word	Page	None	3865	13978	718	2328
4	100	Line	Page	None	3329	5284	777	2473
2	1000	Line	Page	None	2687	4322	660	2095
2	100	Line	Page	4	2417	3328	689	2234
2	100	Line	6144 tokens	None	2377	3920	602	1916

VRDU Registration Form

# Coord.	# Quant.	Segment	Chunking	JSON Indent	Input		Target	
					50^{th}	99^{th}	50^{th}	99^{th}
2	100	Line	Page	None	963	1578	79	147
2	100	Word	Page	None	3083	5196	101	349
4	100	Line	Page	None	1232	2017	91	177
2	1000	Line	Page	None	1052	1723	83	155
2	100	Line	Page	4	977	1592	92	160
2	100	Line	6144 tokens	None	963	1578	79	147

CORD

# Coord.	# Quant.	Segment	Chunking	JSON Indent	Input		Target	
					50^{th}	99^{th}	50^{th}	99^{th}
2	100	Line	Page	None	342	869	355	1495
2	100	Word	Page	None	396	1067	375	1638
4	100	Line	Page	None	408	1139	422	1801
2	1000	Line	Page	None	364	959	376	1957
2	100	Line	Page	4	411	938	474	1997
2	100	Line	6144 tokens	None	342	869	355	1495

```
{
  "advertiser": "",
  "agency": "",
  "contract_num": "",
  "flight_from": "",
  "flight_to": "",
  "gross_amount": "",
  "line_item": [
    {
      "channel": "",
      "program_desc": "",
      "program_end_date": "",
      "program_start_date": "",
      "sub_amount": ""
    }
  ],
  "product": "",
  "property": "",
  "tv_address": ""
}
```


Figure 4: VRDU Ad-Buy Form Schema.


```
{
  "file_date": "",
  "foreign_principle_name": "",
  "registrant_name": "",
  "registration_num": "",
  "signer_name": "",
  "signer_title": ""
}
```


Figure 5: VRDU Registration Form Schema.

```
{
  "line_item": [ # menu
    {
      "discount_price": "", # menu.discountprice
      "identifier": "", # menu.num
      "name": "", # menu.nm
      "other": "", # menu.etc
      "quantity": "", # menu.qty
      "sub_name": [], # menu.sub_nm
      "sub_price": [], # menu.sub_price
      "sub_quantity": [], # menu.sub_qty
      "subtotal_price": "", # menu.itemsubtotal
      "total_price": "", # menu.price
      "unit_price": "" # menu.unitprice
    }
  ],
  "subtotal": { # subtotal
    "discount_price": "", # subtotal.discount_price
    "other": [], # subtotal.etc
    "service_price": "", # subtotal.service_price
    "subtotal_price": [], # subtotal.subtotal_price
    "tax_price": [] # subtotal.tax_price
  },
  "total": { # total
    "cash_price": [], # total.cashprice
    "change_price": "", # total.changeprice
    "credit_card_price": "", # total.creditcardprice
    "emoney_price": "", # total.emoneyprice
    "line_item_quantity_count": "", # total.menuqty_cnt
    "line_item_type_count": "", # total.menutype_cnt
    "other": "", # total.total_etc
    "total_price": [] # total.total_price
  }
}
```

Figure 6: CORD Schema. Note that the original entity types (shown as comments) have been renamed to more semantically meaningful names.

A.6 Zero-shot Baselines Details

We compare LMDX to other Large Model baselines on all benchmarks in the zero-shot context. Those baselines are detailed below.

Text-based Baselines. We evaluate the zero-shot extraction ability of multiple strong Large Language Models: GPT-3.5[3], Gemini Pro (Anil et al., 2023) and PaLM 2-S (Google et al., 2023). To do so, we prompt them with the raw benchmark's OCR text (no coordinate tokens or segment identifier like for LMDX), and extraction instructions alongside the schema in JSON format. We then parse the completions as JSON to get the predicted entities directly. Note that those predicted entities are not localized within the document. A sample prompt can be observed in Figure 7. In particular for GPT-3.5, we use the `gpt-3.5-turbo-1106` through OpenAI's API.

Page Image-based Baselines. We evaluate the zero-shot extraction ability of multiple strong Vision-Language Models: LLaVA-v1.5-13B (Liu et al., 2023), Gemini Pro (Anil et al., 2023), and GPT-4V (OpenAI, 2023a). The prompt includes task description, instructions and target schema represented in JSON format as text input and the document page as image input. We provide examples of valid JSON values in the task instructions. Note that those predicted entities are not localized within the document. A sample prompt can be observed in Figure 8. In particular for GPT-4V, we use the `gpt-4-1106-preview` through OpenAI's API. For Gemini Pro, we use `gemini-pro-vision` through the VertexAI API[4].

A.7 CORD Baselines Details

LayoutLMv3 Baseline. We follow the released implementation[5] for the $\text{LayoutLMv3}_{\text{LARGE}}$ model and the training protocol described in (Huang et al., 2022) as closely as possible. In particular, we train the model for 80 epochs for each experiment on CORD (namely, 10, 50, 100 and 200-document training sets), on the IOB tags of the leaf entities. One difference in our training is that, due to computational resource constraints, we use $batch\_size = 8$ and $learning\_rate = 2 \cdot 10^{-5}$.

As the LayoutLMv3 model can only extract leaf entities, we design and heavily optimize a heuristic algorithm to group the leaf entities into hierarchical entities *menu*, *subtotal* and *total*. The best heuristics we could find are as follows:

- For the *subtotal* and *total* hierarchical entity types, since they appear only once per document, we group all their extracted sub-entities under a single *subtotal* and *total* entity, respectively.

- For *menu* hierarchical entity type, we observe that those entities usually occur multiple times on a document, and each *menu* has at most one *nm*, *num*, *unitprice*, *cnt*, *discountprice*, *price*, *itemsubtotal*, *etc* sub-entities and potentially multiple *sub_nm*, *sub_price* and *sub_cnt* sub-entities. We also notice that the sub-entities aligned horizontally overwhelmingly belong to the same *menu* entity, and a *menu* entity can sometimes span over two or more consecutive horizontal lines. To leverage those observations, we perform a two-step grouping process for *menu* entities. First, we merge the extracted leaf sub-entities into horizontal groups, where a threshold of 0.5 on the intersection-over-union of the Y-axis was used for the determination of horizontal alignment. Second, we further merge the *consecutive* horizontal groups into *menu* entities, if and only if the horizontal groups do not have type duplication in any of the *nm*, *num*, *unitprice*, *cnt*, *discountprice*, *price*, *itemsubtotal*, and *etc* sub-entities (namely, those sub-entities only show up in at most one of the consecutive horizontal groups to be merged). We allow duplication of *sub_nm*, *sub_price* and *sub_cnt* sub-entity types. After those two steps, we obtain the final *menu* entities.

Donut Baseline. We follow Donut released implementation[6] for the Donut benchmarking results on CORD. We use the default training configuration for all experiments on CORD (namely, 10, 50, 100 and 200-document training sets), with the following difference: we reduce batch size from 8 to 4 due to computational resource constraints, and increase the number of train epochs from 30 to 60. For each experiment, checkpoint with the lowest loss on the dev set is selected and we report performance on test set. Micro-F1 scores produced by Donut evaluation code are reported (similar to all our other models).

[3] https://platform.openai.com/docs/models/gpt-3-5-turbo

[4] https://cloud.google.com/vertex-ai/docs/generative-ai/start/quickstarts/quickstart-multimodal

[5] https://github.com/microsoft/unilm/tree/master/layoutlmv3

[6] https://github.com/clovaai/donut

```
${RAW_OCR_TEXT}

Given the document, extract the text value of the entities included in
the schema in json format.
- The extraction must respect the JSON schema.
- Only extract entities specified in the schema. Do not skip any entity types.
- The values must only include text found in the document.
- Use null or [] for missing entity types.
- Do not indent the json you produce.
- Examples of valid string value format: "$ 1234.50", "John Do", null.
- Examples of valid list value format: ["$ 1234.50", "John Do"], [].

Schema: {"file_date": "", "foreign_principle_name": "",
"registrant_name": "", "registration_num": "", "signer_name": "",
"signer_title": ""}
```json
```

Figure 7: Sample prompt for GPT-3.5$_{+OCR}$, Gemini Pro$_{+OCR}$ and PaLM 2-S$_{+OCR}$ text-based baselines for VRDU Registration Form.

```
${DOCUMENT_PAGE_IMAGE}

Given the document, extract the text value of the entities included in
the schema in json format.
- The extraction must respect the JSON schema.
- Only extract entities specified in the schema. Do not skip any entity types.
- The values must only include text found in the document.
- Use null or [] for missing entity types.
- Do not indent the json you produce.
- Examples of valid string value format: "$ 1234.50", "John Do", null.
- Examples of valid list value format: ["$ 1234.50", "John Do"], [].

Schema: {"file_date": "", "foreign_principle_name": "",
"registrant_name": "", "registration_num": "", "signer_name": "",
"signer_title": ""}
```json
```

Figure 8: Sample prompt for GPT-4V$_{+Image}$, LLaVA-v1.5-13B$_{+Image}$, and Gemini Pro$_{+Image}$ image-based baselines for VRDU Registration Form.

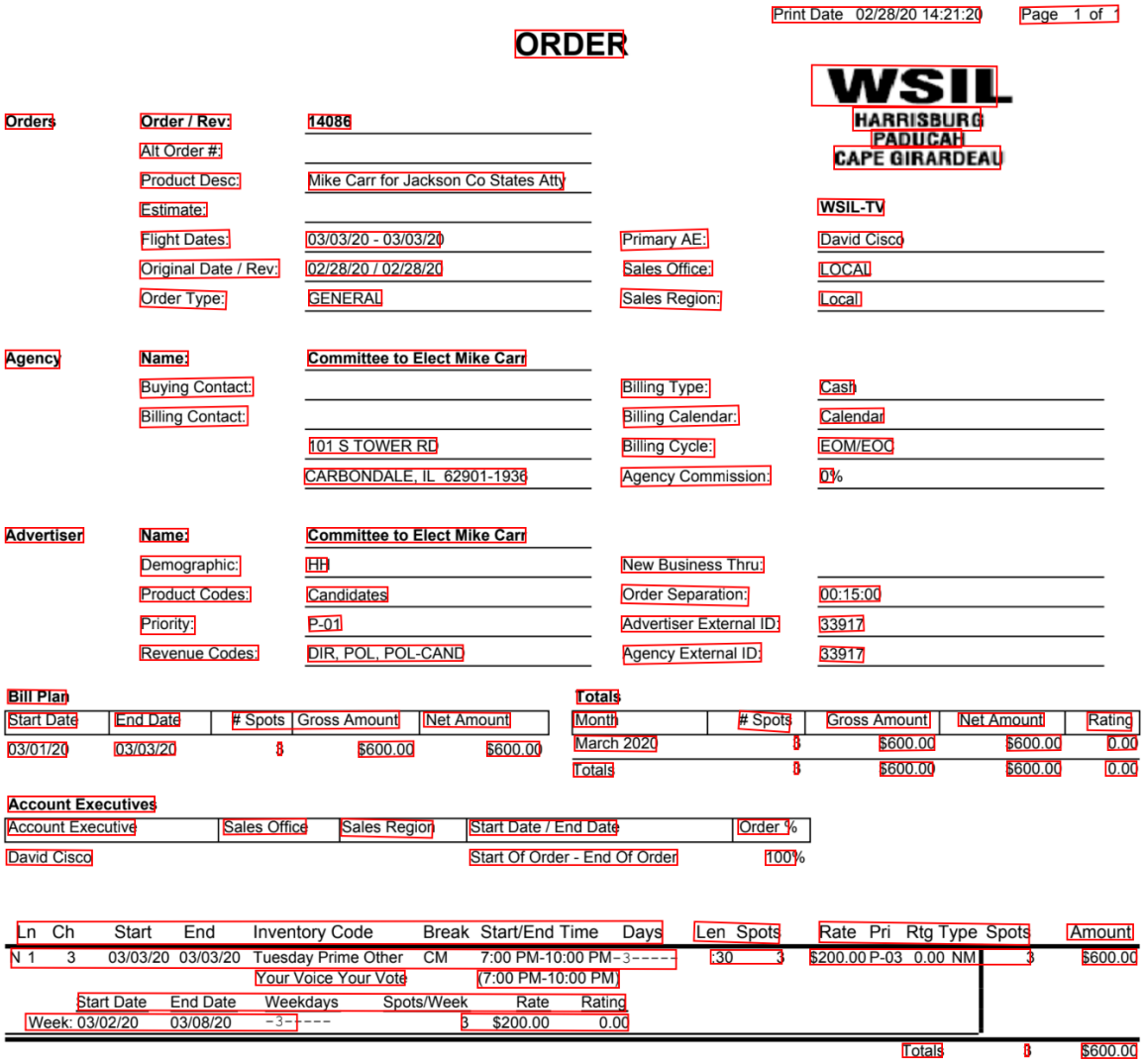
A.8 Sample Prompts and Completions

In this section, we present example of LMDX prompts and completions from the LLM on the VRDU Ad-Buy dataset to better showcase the format used. Figure 9 shows the original document with the line bounding boxes from OCR, Figure 10 shows the corresponding prompt and completion on that document with coordinate segment identifiers, Figure 11 shows the same prompt and completion, but with line index segment identifiers (used in ablation studies to showcase how the LLM can interpret the layout) and Figure 12 shows how the completion changes when skipping missing entity types entirely in the completion. Finally, Figure 13 and Figure 14 show how the prompts/completions changes if 4 line-level $[x_{\text{min}}, y_{\text{min}}, x_{\text{max}}, y_{\text{max}}]$ coordinates or 2-word level $[x_{\text{center}}, y_{\text{center}}]$ coordinates are used.

A.9 Nearest Neighbors In-Context Learning

In our study, nearest neighbors leads to a significant quality gain over randomly selecting examplars. In this section, we explore why that is the case in the context of VRD information extraction. Figures 15, 16 and 17 show typical retrievals using sentenceT5 (Ni et al., 2021) embeddings[7] on the OCR text for similarity. Unsurprisingly, nearest neighbors works well as it retrieves examplars from the same template as the target document, i.e. from the same merchant in the case of CORD documents (store/restaurant receipts). As those examples share the same layout, boilerplate text, and entities, it makes it easier for the model to understand the correct extraction pattern, leading to significant gains in quality.

[7] https://www.kaggle.com/models/google/sentence-t5/frameworks/tensorFlow2/variations/st5-base



Print Date 02/28/20 14:21:20 Page 1 of 1

ORDER

WSIL
HARRISBURG
PADUCAH
CAPE GIRARDEAU

Orders	**Order / Rev:**	**14086**		
	Alt Order #:			
	Product Desc:	Mike Carr for Jackson Co States Atty		
	Estimate:			**WSIL-TV**
	Flight Dates:	03/03/20 - 03/03/20	Primary AE:	David Cisco
	Original Date / Rev:	02/28/20 / 02/28/20	Sales Office:	LOCAL
	Order Type:	GENERAL	Sales Region:	Local
Agency	**Name:**	**Committee to Elect Mike Carr**		
	Buying Contact:		Billing Type:	Cash
	Billing Contact:		Billing Calendar:	Calendar
		101 S TOWER RD	Billing Cycle:	EOM/EOQ
		CARBONDALE, IL 62901-1936	Agency Commission:	0%
Advertiser	**Name:**	**Committee to Elect Mike Carr**		
	Demographic:	HH	New Business Thru:	
	Product Codes:	Candidates	Order Separation:	00:15:00
	Priority:	P-01	Advertiser External ID:	33917
	Revenue Codes:	DIR, POL, POL-CAND	Agency External ID:	33917

Bill Plan

Start Date	End Date	# Spots	Gross Amount	Net Amount
03/01/20	03/03/20	3	$600.00	$600.00

Totals

Month	# Spots	Gross Amount	Net Amount	Rating
March 2020	3	$600.00	$600.00	0.00
Totals	3	$600.00	$600.00	0.00

Account Executives

Account Executive	Sales Office	Sales Region	Start Date / End Date	Order %
David Cisco			Start Of Order - End Of Order	100%

Ln	Ch	Start	End	Inventory Code	Break	Start/End Time	Days	Len	Spots	Rate	Pri	Rtg Type	Spots	Amount
N 1	3	03/03/20	03/03/20	Tuesday Prime Other	CM	7:00 PM-10:00 PM	-3-----	:30	3	$200.00	P-03	0.00 NM	3	$600.00
				Your Voice Your Vote		(7:00 PM-10:00 PM)								

Start Date	End Date	Weekdays	Spots/Week	Rate	Rating
Week: 03/02/20	03/08/20	-3-----	3	$200.00	0.00

Totals 3 $600.00

Figure 9: Example document from VRDU Ad-Buy Form. The line bounding boxes detected by OCR are shown in red.

```
========================================== Prompt ==========================================

<Document>
Print Date 02/28/20 14:21:20 75|02
Page 1 of 1 91|02
ORDER 49|04
WSIL 77|07
Orders 04|09
Order/Rev: 17|09
14086 29|09
HARRISBURG 78|09
PADUCAH 78|10
CAPE GIRARDEAU 78|12
Alt Order #: 17|11
Product Desc: 18|13
Mike Carr for Jackson Co States Atty 38|13
Estimate: 16|15
WSIL-TV 73|15
Flight Dates: 17|17
03/03/20 - 03/03/20 33|17
Primary AE: 57|17
David Cisco 73|17
Original Date/ Rev: 19|19
02/28/20/02/28/20 33|19
...
Start Date End Date Weekdays Spots/Week Rate Rating 31|66
Week: 03/02/20 03/08/20 -3-- 15|67
3 $200.00 0.00 47|67
Totals 79|69
3 87|69
$600.00 94|69
</Document>
<Task>
From the document, extract the text values and tags of the following
entities:
{"advertiser": "", "agency": "", "contract_num": "", "flight_from": "",
"flight_to": "", "gross_amount": "", "line_item": [{"channel": "",
"program_desc": "", "program_end_date": "", "program_start_date": "",
"sub_amount": ""}], "product": "", "property": "", "tv_address": ""}
</Task>
<Extraction>



========================================= Completion =========================================

{"advertiser": "Committee to Elect Mike Carr 37|36", "agency":
"Committee to Elect Mike Carr 37|24", "contract_num": "14086 29|09",
"flight_from": "03/03/20 33|17", "flight_to": "03/03/20 33|17",
"gross_amount": "$600.00 77|51", "line_item": [{"channel": "3 31|63",
"program_desc": "Tuesday Prime 31|63\nYour Voice Your Vote 29|64",
"program_end_date": "03/03/20 31|63", "program_start_date":
"03/03/20 31|63", "sub_amount": "$600.00 94|63"}], "product":
"Mike Carr for Jackson Co States Atty 38|13", "property": "WSIL 77|07",
"tv_address": null}
</Extraction>
```

Figure 10: VRDU Ad-Buy Form sample prompt and completion with 2 $[x_{\text{center}}, y_{\text{center}}]$ coordinates for segment identifier. The document is truncated for easier visualization.

```
========================================== Prompt ==========================================

<Document>
Print Date 02/28/20 14:21:20 <0>
Page 1 of 1 <1>
ORDER <2>
WSIL <3>
Orders <4>
Order/Rev: <5>
14086 <6>
HARRISBURG <7>
PADUCAH <8>
CAPE GIRARDEAU <9>
Alt Order #: <10>
Product Desc: <11>
Mike Carr for Jackson Co States Atty <12>
Estimate: <13>
WSIL-TV <14>
Flight Dates: <15>
03/03/20 - 03/03/20 <16>
Primary AE: <17>
David Cisco <18>
Original Date/ Rev: <19>
02/28/20/02/28/20 <20>
...
Start Date End Date Weekdays Spots/Week Rate Rating <105>
Week: 03/02/20 03/08/20 -3-- <106>
3 $200.00 0.00 <107>
Totals <108>
3 <109>
$600.00 <110>
</Document>
<Task>
From the document, extract the text values and tags of the following
entities:
{"advertiser": "", "agency": "", "contract_num": "", "flight_from": "",
"flight_to": "", "gross_amount": "", "line_item": [{"channel": "",
"program_desc": "", "program_end_date": "", "program_start_date": "",
"sub_amount": ""}], "product": "", "property": "", "tv_address": ""}
</Task>
<Extraction>



======================================== Completion ========================================

{"advertiser": "Committee to Elect Mike Carr <44>", "agency":
"Committee to Elect Mike Carr <29>", "contract_num": "14086 <6>",
"flight_from": "03/03/20 <16>", "flight_to": "03/03/20 <16>",
"gross_amount": "$600.00 <84>", "line_item": [{"channel": "3 <102>",
"program_desc": "Tuesday Prime Other CM <102>", "program_end_date":
"03/03/20 <102>", "program_start_date": "03/03/20 <102>",
"sub_amount": "$600.00 <100>"}], "product": "Mike Carr for Jackson Co
States Atty <12>", "property": "WSIL-TV <14>", "tv_address": null}
</Extraction>
```

Figure 11: VRDU Ad-Buy Form sample prompt and completion with line index for segment identifier. Instead of using coordinate tokens, the line index is given as <line_index> at the end of each line, which does not communicate layout information but still uniquely identify the segment. The document is truncated for easier visualization.

```
========================================== Prompt ==========================================

<Document>
Print Date 02/28/20 14:21:20 75|02
Page 1 of 1 91|02
ORDER 49|04
WSIL 77|07
Orders 04|09
Order/Rev: 17|09
14086 29|09
HARRISBURG 78|09
PADUCAH 78|10
CAPE GIRARDEAU 78|12
Alt Order #: 17|11
Product Desc: 18|13
Mike Carr for Jackson Co States Atty 38|13
Estimate: 16|15
WSIL-TV 73|15
Flight Dates: 17|17
03/03/20 - 03/03/20 33|17
Primary AE: 57|17
David Cisco 73|17
Original Date/ Rev: 19|19
02/28/20/02/28/20 33|19
...
Start Date End Date Weekdays Spots/Week Rate Rating 31|66
Week: 03/02/20 03/08/20 -3-- 15|67
3 $200.00 0.00 47|67
Totals 79|69
3 87|69
$600.00 94|69
</Document>
<Task>
From the document, extract the text values and tags of the following
entities:
{"advertiser": "", "agency": "", "contract_num": "", "flight_from": "",
"flight_to": "", "gross_amount": "", "line_item": [{"channel": "",
"program_desc": "", "program_end_date": "", "program_start_date": "",
"sub_amount": ""}], "product": "", "property": "", "tv_address": ""}
</Task>
<Extraction>



======================================== Completion ========================================

{"advertiser": "Committee to Elect Mike Carr 37|36", "agency":
"Committee to Elect Mike Carr 37|24", "contract_num": "14086 29|09",
"flight_from": "03/03/20 33|17", "flight_to": "03/03/20 33|17",
"gross_amount": "$600.00 77|51", "line_item": [{"channel": "3 31|63",
"program_desc": "Tuesday Prime 31|63\nYour Voice Your Vote 29|64",
"program_end_date": "03/03/20 31|63", "program_start_date":
"03/03/20 31|63", "sub_amount": "$600.00 94|63"}], "product":
"Mike Carr for Jackson Co States Atty 38|13", "property": "WSIL 77|07"}
</Extraction>
```

Figure 12: VRDU Ad-Buy Form sample prompt and completion with skipping the missing entity types in the completion (tv_address in this case). Instead of explicitly outputting the missing entity type `"tv_address": null`, the entity type is skipped entirely. The document is truncated for easier visualization.

```
========================================== Prompt ==========================================

<Document>
Print Date 02/28/20 14:21:20 66|02|83|03
Page 1 of 1 87|02|95|03
ORDER 45|03|54|05
WSIL 69|06|85|08
Orders 02|09|06|10
Order/Rev: 14|09|21|10
14086 28|09|31|10
HARRISBURG 73|08|83|10
PADUCAH 74|10|82|11
CAPE GIRARDEAU 71|11|85|12
Alt Order #: 14|11|20|12
Product Desc: 14|12|22|13
Mike Carr for Jackson Co States Atty 28|12|49|14
Estimate: 14|14|19|15
WSIL-TV 70|14|75|15
Flight Dates: 14|16|21|17
03/03/20 - 03/03/20 27|16|39|17
Primary AE: 53|16|61|17
David Cisco 70|16|77|17
Original Date/ Rev: 14|18|25|19
02/28/20/02/28/20 27|18|39|19
...
Start Date End Date Weekdays Spots/Week Rate Rating 08|65|54|66
Week: 03/02/20 03/08/20 -3-- 04|66|27|67
3 $200.00 0.00 40|66|54|67
Totals 77|68|80|69
3 87|68|87|69
$600.00 92|68|96|69
</Document>
<Task>
From the document, extract the text values and tags of the following
entities:
{"advertiser": "", "agency": "", "contract_num": "", "flight_from": "",
"flight_to": "", "gross_amount": "", "line_item": [{"channel": "",
"program_desc": "", "program_end_date": "", "program_start_date": "",
"sub_amount": ""}], "product": "", "property": "", "tv_address": ""}
</Task>
<Extraction>



======================================== Completion ========================================

{"advertiser": "Committee to Elect Mike Carr 27|35|46|36",
"agency": "Committee to Elect Mike Carr 28|24|46|25", "contract_num": "14086 28|09|31|10",
"flight_from": "03/03/20 - 03/03/20 27|16|39|17",
"flight_to": "03/03/20 - 03/03/20 27|16|39|17", "gross_amount": null,
"line_item": [{"channel": "N 1 3 0 ... Tuesday Prime Other CM 7:00 PM-10:00 PM-3----- 03|62|58|63",
"program_desc": "Your Voice Your Vote 23|64|36|64",
"program_end_date": "03/03/20 12|49|17|50", "program_start_date": "03/03/20 03|49|08|50",
"sub_amount": null}], "product": "WSIL-TV 70|14|75|15", "property": null, "tv_address": null}
</Extraction>
```

Figure 13: VRDU Ad-Buy Form sample prompt and completion with 4 line-level $[x_{\text{min}}, y_{\text{min}}, x_{\text{max}}, y_{\text{max}}]$ coordinates in the prompt and completion. The document and completion is truncated for easier visualization.

```
========================================== Prompt ==========================================

<Document>
Print 67|02 Date 70|02 02/28/20 75|02 14:21:20 81|02 Page 88|02 1 91|02 of 93|02 1 95|02
ORDER 49|04 WSIL 77|07 Orders 04|09 Order 15|09 / 18|09 Rev 19|09 : 21|09 14086 29|09
HARRISBURG 78|09 PADUCAH 78|10 CAPE 73|12 GIRARDEAU 80|12 Alt 14|11 Order 17|11 # 19|11
: 20|11 Product 16|13 Desc 20|13 : 22|13 Mike 29|13 Carr 32|13 for 34|13 Jackson 38|13
Co 41|13 States 45|13 Atty 48|13 Estimate 16|15 : 19|15 WSIL 71|15 - 74|15 TV 75|15
Flight 15|17 Dates 19|17 : 21|17 03/03/20 30|17 - 33|17 03/03/20 36|17 Primary 56|17
AE 59|17 : 61|17 David 71|17 Cisco 75|17 Original 16|19 Date 20|19 / 22|19 Rev 23|19
: 25|19 02/28/20/02/28/20 33|19 Sales 55|19 Office 59|19 : 61|19 LOCAL 72|19 Order 15|21
Type 19|21 : 21|21 GENERAL 31|20 Sales 55|21 Region 59|21 : 62|21 Local 72|21 Agency 05|24
Name 15|24 : 17|24 Committee 31|24 to 35|24 Elect 38|24 Mike 41|24 Carr 44|24 Cash 71|26
Buying 16|26 Contact 20|26 : 23|26 Billing 15|28 Contact 20|28 : 22|28 Billing 55|26
Type 59|26 : 61|26 Billing 55|28 Calendar 60|28 : 63|28 Calendar 73|28 101 29|30 S 30|30
TOWER 34|30 RD 37|30 EOM 71|30 / 73|30 EOC 75|30 Billing 55|30 Cycle 59|30 : 61|30
Agency 56|32 Commission 62|32 : 66|32 CARBONDALE 32|32 , 36|32 IL 37|32 62901-1936 42|32
0 70|32 % 71|32 Advertiser 06|36 Name 15|36 : 17|36 Committee 31|36 to 35|36 Elect 38|36
Mike 41|36 Carr 44|36 Demographic 17|38 : 22|38 HH 28|38 New 55|38 Business 59|38
Thru 64|38 : 65|38 Product 16|39 Codes 20|39 : 23|39 Candidates 31|39 Order 55|39
Separation 60|40 : 64|40 00:15:00 73|39 Priority 16|41 : 18|41 P 28|41 - 29|41 01 30|41
Advertiser 57|41 External 62|41 ID 65|41 : 67|41 33917 72|41 Revenue 16|43 Codes 21|43 :
23|43 DIR 28|43 , 30|43 POL 32|43 , 33|43 POL 35|43 - 37|43 CAND 39|43 Agency 56|43
External 61|43 ID 64|43 : 65|43 33917 72|43 Bill 04|46 Plan 06|46 Start 04|48 Date 07|48
End 13|47 Date 16|47 # 21|48 Spots 24|48 Gross 28|48 Amount 33|48 Net 38|47 Amount 42|48
Totals 52|46 Month 51|47 March 51|49 2020 55|49 # 64|47 Spots 66|48 Net 83|47 Amount 86|47
Gross 72|48 Amount 77|48 $ 75|49 600.00 78|49 Rating 94|48 0.00 95|49 03/01/20 05|49
03/03/20 14|49 3 25|49 3 68|49 $ 32|49 600.00 34|49 $ 43|49 600.00 45|49 $ 86|49
600.00 88|49 Totals 51|51 3 68|51 $ 75|51 600.00 77|51 $ 86|51 600.00 88|51 0.00 95|51
Account 05|53 Executives 12|53 Account 05|54 Executive 10|54 Sales 22|54 Office 26|54
Sales 32|54 Region 36|54 Start 42|54 Date 45|54 / 47|54 End 49|54 Date 52|54 Order 65|54
% 67|54 David 04|56 Cisco 08|56 Start 42|56 Of 45|56 Order 48|56 - 50|56 End 51|56 Of 54|56
Order 57|56 100 67|56 % 68|56 Len 61|61 Spots 65|61 : 61|63 30 62|63 3 67|63 Rate 71|61 Pri
75|61 Rtg 78|61 Type 81|61 Spots 86|61 $ 69|63 200.00 72|63 P 74|63 - 75|63 03 76|63
0.00 79|63 NM 82|63 3 87|63 Amount 93|61 $ 92|63 600.00 94|63 Ln 04|61 Ch 07|61 Start 13|61
End 18|61 Inventory 26|61 Code 31|61 Break 39|61 Start 10|65 Date 13|65 End 17|66 Date 20|66
Weekdays 27|66 Spots 35|66 / 37|66 Week 39|66 Rate 46|66 Rating 52|66 Week 06|67 : 08|67
03/02/20 11|67 03/08/20 18|67 -3-- 25|67 3 40|67 $ 43|67 200.00 45|67 0.00 53|67 Totals 79|69
3 87|69 $ 92|69 600.00 94|69
</Document>
<Task>
From the document, extract the text values and tags of the following
entities:
{"advertiser": "", "agency": "", "contract_num": "", "flight_from": "",
"flight_to": "", "gross_amount": "", "line_item": [{"channel": "",
"program_desc": "", "program_end_date": "", "program_start_date": "",
"sub_amount": ""}], "product": "", "property": "", "tv_address": ""}
</Task>
<Extraction>



========================================== Completion ==========================================

{"advertiser": "Committee to Elect Mike Carr 44|24", "agency": "Cash 71|26",
"contract_num": "14086 29|09", "flight_from": "03/03/20 30|17", "flight_to": "03/03/20 36|17",
"gross_amount": "600.00 77|49", "line_item": [{"channel": "1 03|63",
"program_desc": "Your Voice Your Vote 34|64", "program_end_date": "03/03/20 13|63",
"program_start_date": "03/03/20 19|63", "sub_amount": "600.00 92|63"}],
"product": "WSIL 71|15 - 74|15 TV 75|15", "property": "WSIL 71|15 - 74|15 TV 75|15",
"tv_address": null}
</Extraction>
```

Figure 14: VRDU Ad-Buy Form sample prompt and completion with 2 word-level $[x_{\text{center}}, y_{\text{center}}]$ coordinates in the prompt and completion. With word-level coordinate tokens, the text sequence of the prompt is mostly composed of coordinate tokens, and becomes far from the usual text sequences LLMs are trained on.

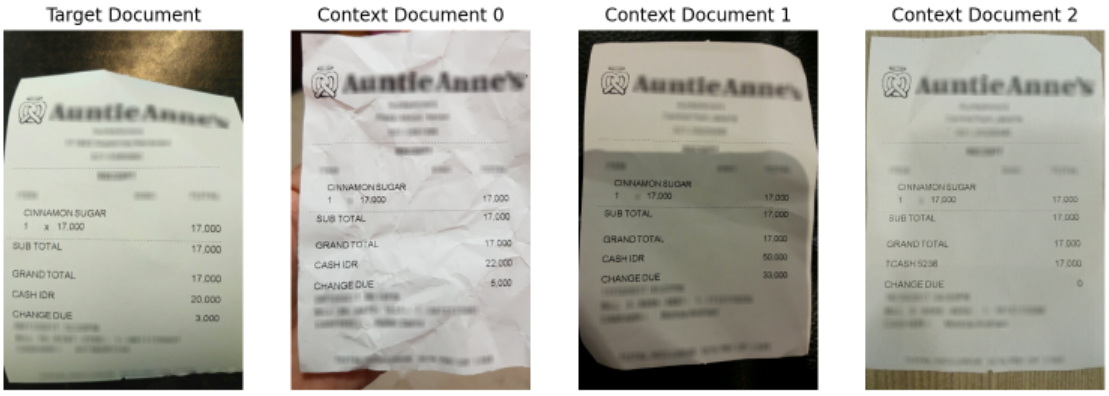


Figure 15: Nearest Neighbors on CORD, Example 1, retrieving examplars from the same merchant.

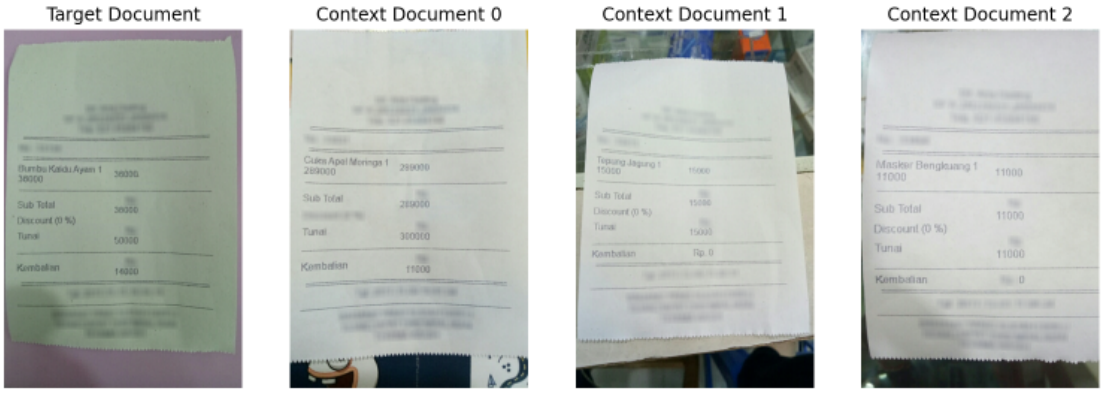


Figure 16: Nearest Neighbors on CORD, Example 2, retrieving examplars from the same merchant.

Figure 17: Nearest Neighbors on CORD, Example 3, retrieving examplars from the same merchant.

Table 7: Parsing error rates from $\text{LMDX}_{\text{PaLM 2-S}}$ responses on VRDU Ad-Buy Mixed and CORD datasets.

$\lvert\mathcal{D}\rvert$	Dataset	Invalid JSON	Invalid Entity Value Format	Entity Text Not Found
0	Ad-Buy	0.18%	0.04%	0.59%
	CORD	0.00%	0.00%	0.00%
10	Ad-Buy	0.27%	0.04%	0.44%
	CORD	0.00%	0.00%	0.00%
50	Ad-Buy	0.24%	0.00%	0.17%
	CORD	0.06%	0.00%	0.00%
100	Ad-Buy	0.24%	0.00%	0.13%
	CORD	0.00%	0.03%	0.00%
200	Ad-Buy	0.25%	0.00%	0.09%
	CORD	0.00%	0.00%	0.00%

A.10 Completion Parsing Error Rates

In this section, we report the various completion parsing error types and their occurrence rates for $\text{LMDX}_{\text{PaLM 2-S}}$.

Invalid JSON Formatting. This refers to cases for which Python's `json.loads(completion)` fails on a LLM's completion. As observed in Table 7, the JSON parsing error rate is below 0.3% in all training settings.

Invalid Entity Value Format. This refers to cases where the leaf entity value does not follow the expected `"<text-segment-1> XX|YY <text-segment-2> XX|YY"` format. As observed in Table 7, the Invalid Entity Value Format Rate is below 0.05% in all training settings.

Hallucination / Entity Text Not Found. This refers to cases where the segment identifier is valid, but the entity text does not appear on the predicted segment (hallucination). As observed in Table 7, the Entity Text Not Found error rate is below 0.6% in all training settings. As part of LMDX methodology, we discard any prediction whose text does not appear on the specified segment, ensuring we discard all hallucination.

Note that those numbers are computed at the completion level. Since multiple completions are sampled for each document chunk, the sampling scheme allows for correcting those errors and no document in the benchmarks fail extraction.

A.11 Error Analysis

In this section, we perform an error analysis on the test set to identify common error patterns of LMDX. A very common error type we observe is caused by OCR lines grouping multiple semantically different segments. We show two instance of those cases observed in $\text{LMDX}_{\text{PaLM 2-S}}$ on the VRDU Ad-Buy Form in Figure 18. In the first example, prediction for the entity *line_item/program_desc* includes text from the previous column "Channel" along with the value in the column "Description". From the OCR line bounding boxes, we can see that these two columns are grouped as the same OCR line. In the second example, the model confuses between the adjacent keys "Invoice Period" and "Flight Dates" and extracts invoice dates as flight dates. Similar to the first example, OCR line bounding boxes show that the invoice dates and the key "Flight Dates" are grouped together in the same line although they are semantically different. As $\text{LMDX}_{\text{PaLM 2-S}}$ uses only coarse line layout information ($[\text{x}_{\text{center}}, \text{y}_{\text{center}}]$ with 100 quantization buckets), the model fails in these cases, which is a current limitation of LMDX. We believe that incorporating the image modality will make LMDX more robust to those OCR errors.

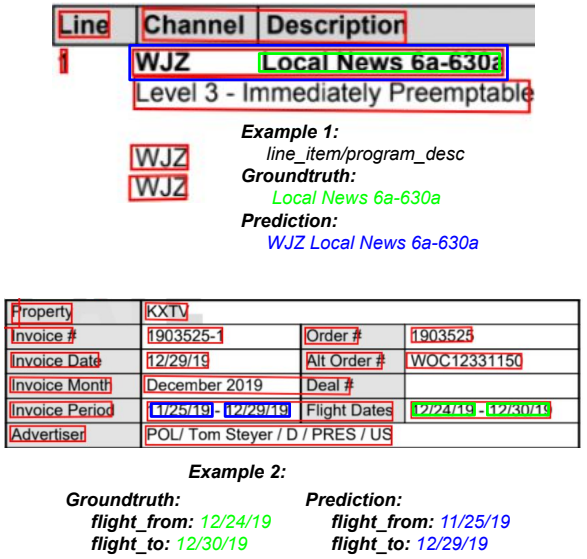


Figure 18: Typical error pattern of $\text{LMDX}_{\text{PaLM 2-S}}$. In both examples, the detected OCR lines are shown in red, the model predicted entities are shown in blue, and the groundtruth entities are shown in green. In both cases, the detected OCR lines merge two semantically distinct segments, causing the model to wrongly associate them in its predictions.

A.12 Effect of Coordinate Tokenization Schemes

In this section, we study how different designs of coordinate tokens affect extraction quality, to determine which is the most effective. There are many ways to tokenize the coordinates of the text segments, e.g. using line versus words as segment, the number of coordinates to use per segment ($[x_{\text{center}}, y_{\text{center}}]$ versus $[x_{\text{min}}, y_{\text{min}}, x_{\text{max}}, y_{\text{max}}]$). To establish which

scheme is the most effective, we evaluate the zero-shot performance of $\text{LMDX}_{\text{PaLM 2-S}}$ on VRDU Registration Form, Ad-Buy Form and CORD benchmarks with the following schemes:

- 2 Line-level $[x_{\text{center}}, y_{\text{center}}]$ coordinates with $B = 100$ quantization buckets.
- 4 Line-level $[x_{\text{min}}, y_{\text{min}}, x_{\text{max}}, y_{\text{max}}]$ coordinates with $B = 100$ quantization buckets.
- 2 Word-level $[x_{\text{center}}, y_{\text{center}}]$ coordinates with $B = 100$ quantization buckets.

An example of prompt and completion for each scheme is given in Appendix A.8. Results quality is given in Figure 19.

On all benchmarks, 2-line level $[x_{\text{center}}, y_{\text{center}}]$ coordinates obtains the best performance. This is caused by that coordinate tokenization scheme being very token efficient (see Appendix A.4). 2 Word-level $[x_{\text{center}}, y_{\text{center}}]$ coordinates and 4 Line-level $[x_{\text{min}}, y_{\text{min}}, x_{\text{max}}, y_{\text{max}}]$ coordinates increase the tokens used by coordinates drastically, leading to multiple chunks generated for each document page, hence lowering quality since the model won't be able to correctly predict entities spanning multiple chunks. Furthermore, the large number of coordinate tokens added by those schemes makes the text sequences used in LMDX significantly different from the natural text sequences the LLM was pretrained on, hence making the LLM less effective at interpreting them, thereby lowering extraction quality.

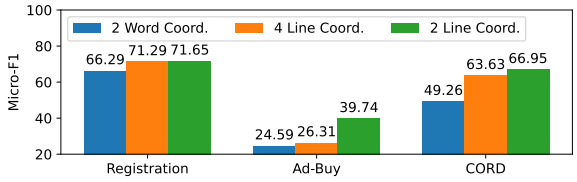


Figure 19: Zero-shot performance of $\text{LMDX}_{\text{PaLM 2-S}}$ on VRDU Registration Form Mixed Template, Ad-Buy Form Mixed Template, and CORD with different coordinate tokenization schemes. On all benchmarks, 2-line level $[x_{\text{center}}, y_{\text{center}}]$ coordinates obtains the best performance.

A.13 Latency Comparison

In Table 8, we compare the latencies of LMDX and other methodologies, LayoutLMv3 and Donut. Overall, LayoutLMv3 is the fastest model, as it is encoder-only. $\text{LMDX}_{\text{Gemini Pro}}$ is the slowest among accelerated models, due to its reliance on an LLM with long inputs and outputs. Nonetheless, its latency remains acceptable in a production setting.

Table 8: Median, 95$^{\text{th}}$ and 99$^{\text{th}}$ percentile latencies of information extraction solutions on CORD. For text-based solutions LayoutLMv3 and $\text{LMDX}_{\text{Gemini Pro}}$, the reported numbers do not include the OCR latency.

Model	Hardware	Latency (ms)		
		Median	95$^{\text{th}}$ %	99$^{\text{th}}$ %
LayoutLMv3	CPU	648	999	1394
	GPU T4	30	55	93
Donut	CPU	14392	19427	20065
	GPU T4	620	1230	1683
$\text{LMDX}_{\text{Gemini Pro}}$	TPU	3653	7102	8345

A.14 Ablation Detailed Metrics.

In Table 9, we present the per-entity F1 score for the coordinate ablations to highlight which entity most benefit from the introduction of layout information, and explain the large difference observed.

Table 9: Ablation of coordinates for $\text{LMDX}_{\text{PaLM 2-S}}$, where per-entity F1 scores are shown. Adding coordinates greatly increase the extraction quality of entity types that require the understanding of spatial alignment, such as *line_item*.

Entity	Hierar-chical?	Number of occurrence	With Coord.	Without Coord.	Δ
advertiser	✗	635	**95.90**	88.98	-6.92
agency	✗	283	**73.05**	71.84	-1.21
contract_num	✗	624	**78.37**	74.00	-4.37
flight_from	✗	540	**67.74**	63.63	-4.11
flight_to	✗	538	**75.57**	70.38	-5.19
gross_amount	✗	629	**98.86**	98.47	-0.39
product	✗	607	**87.86**	75.33	-12.53
tv_address	✗	535	**81.04**	78.13	-2.91
property	✗	595	**77.75**	74.33	-3.42
line_item	✓	9163	**39.35**	18.35	-21.00

Overall, performance is lower across the board without coordinates, as the VRDU Ad-Buy form benchmark consists entirely of invoices, a very layout-heavy document, where the alignment of the key segments (e.g. "Contract Number: ", "Total Due:", "Product:") and value segments (e.g. "123456", "$1000", "Political Advertisement") matters a lot for the correct understanding of most entities. Thus, removing the layout modality will have a stark difference in performance. However, not all entities are affected in the same way:

- 8 out of 9 leaf entities see single digit drops, with gross_amount affected the least. *gross_amount* can be derived mostly without

looking at its key (e.g."Amount Due") and simply using cues commonly found in money amount entities (e.g. "$", "USD", etc).

- *line_item*, the hierarchical entity in Ad-Buy form, is affected the most. This is expected as *line_item*'s components (*channel*, *program_desc*, *program_start_date*, *program_end_date*, *sub_amount*) are always visually arranged in tables, hence understanding horizontal and vertical alignments of the different text lines (which the coordinates provide) are critical for the correct grouping of those components into *line_items*.
- Since *line_items* are by far the most common entity in VRDU Ad-Buy form (9163 occurrences), they have the most effect on the overall micro-F1.

A.15 Datasets Details

As part of this work, we used the following datasets:

- *VRDU* (Wang et al., 2023d) consists of two public visually-rich documents information extraction benchmarks: *Registration Form*, which consists of 1915 documents recording foreign agents' activities with the US government requiring public disclosure, and *Ad-Buy Form*, which consists of 641 signed receipts between a TV station and a political campaign group, requiring public disclosure to the Federal Communications Commission. All data is in English, and publicly available for research purposes.
- *CORD* (Park et al., 2019) consists of 1000 Indonesian receipts from various shops and restaurants, on which the authors de-identified identifiable information. The data is a mix of Indonesian and English languages, and is publicly available for research purposes.
- *Payment* (Majumder et al., 2020) is a dataset of 14,237 single-page invoices, annotated with Each invoice is from a different vendor, so the documents do not share any common templates. All the data is in English.
- *Synthetic Forms*. Finally, we generated synthetic training data using blank PDF form templates crawled from government websites, and filled with fully synthetic values using an internal tool. All generated data is in English.

All extraction benchmarks we used are publicly available for research purposes, and we limited their usage to this research work. We also checked these datasets to ensure that no personally identifiable information is involved.