LAPDoc: Layout-Aware Prompting for Documents

Marcel Lamott[1][0009−0009−4345−6888], Yves-Noel Weweler[2], Adrian Ulges[1], Faisal Shafait[3], Dirk Krechel[1][0000−0003−0984−5918], and Darko Obradovic[2]

[1] RheinMain University of Applied Sciences, Wiesbaden, Germany
{marcel.lamott**, adrian.ulges, dirk.krechel}@hs-rm.de
[2] Insiders Technologies GmbH, Kaiserslautern, Germany
{y.weweler, d.obradovic}@insiders-technologies.de
[3] National University of Sciences and Technology, Islamabad, Pakistan
faisal.shafait@seecs.edu.pk

Abstract. Recent advances in training large language models (LLMs) using massive amounts of solely textual data lead to strong generalization across many domains and tasks, including document-specific ones. On the other hand, there is a trend to train multi-modal transformer architectures tailored for document understanding that are designed specifically to fuse textual inputs with the corresponding document layout. This involves a separate fine-tuning step for which additional training data is required. At present, no document transformers with comparable generalization to LLMs are available. This raises the question which type of model is to be preferred for document understanding tasks. In this paper we investigate the possibility to use purely text-based LLMs for document-specific tasks by using layout enrichment. We explore drop-in modifications and rule-based methods to enrich purely textual LLM prompts with layout information. In our experiments we investigate the effects on the commercial ChatGPT model and the open-source LLM Solar. We demonstrate that using our approach both LLMs show improved performance on various standard document benchmarks. In addition, we study the impact of noisy OCR and layout errors, as well as the limitations of LLMs when it comes to utilizing document layout. Our results indicate that layout enrichment can improve the performance of purely text-based LLMs for document understanding by up to 15%, and by 6% on average compared to just using plain document text. In conclusion, this approach should be considered for the best model choice between text-based LLM or multi-modal document transformers.

Keywords: Document Understanding · Large Language Models · Layout Understanding · Prompt Enrichment

1 Introduction

In today's business environment, companies face the problem of an ever-growing amount of digital documents that need to be processed. The possibilities of smart

** Corresponding author

devices to capture documents has lead to new digital business models that make heavy use of camera captures, while promising high automation. This induces a dramatic growth in digitized documents of varying quality that need to be processed. In addition to document types such as invoices, forms, complaints, receipts and notices, other, less standardized types of documents are increasingly important, such as contract documents, business reports or legal texts.

Understanding documents completely necessitates understanding of textual and visual modalities as well as the comprehension of the spatial relations between the document's content elements, which guide the reading process and are essential for interpretation [1, p. 1]. Recently, automated document image understanding has taken strides forward: (i) Larger-scale benchmarks that align with real applications [1, 2, 3, 4] allow for real world evaluation and training. (ii) Self-supervised pre-training tasks – with which large amounts of data can be leveraged without the need for hand-crafted annotations – have led to multi-modal neural models. These models can either take document images and text (e.g., extracted by OCR) as input [5, 6, 7, 8], or can operate end-to-end from a purely visual input, essentially also learning OCR in the process [9, 10].

One of the most prominent recent developments in the field of AI has been the rise of large language models (LLMs) such as OpenAI's ChatGPT [11]. These models have been found to excel at various natural language understanding tasks, and have been instruction-tuned to serve as open-domain problem solvers. Key to their success is their scale – with large-scale training data and billions of parameters – which leads to impressive capabilities [12]. In contrast to the aforementioned multi-modal document comprehension models, traditional LLMs process only text[4]. By that the modality of spatial layout, which seems vital for the processing of documents [6, 8], is partially lost due to its reduction to a text sequence.

In this study we focus on an LLM-centric document comprehension pipeline that fuses the text with document layout. First, a document's content is extracted with OCR, resulting in a set of words equipped with box geometries. Second, this information is packaged into a purely textual representation that encodes both the document's text and its spatial structure. We will refer to this step as "verbalization" in the following. Third, the resulting verbalized document is combined with the task description, resulting in a prompt for a pre-trained generative LLM, which solves the document comprehension task at hand without further fine-tuning.

This pipeline offers two *benefits*: First, it exploits the superior knowledge capacity and reasoning capabilities of LLMs – which at the present time have been trained at larger scale and offer larger parametric capacity compared to current multi-modal document-specific models. Second – which is particularly relevant for practical applications – the pipeline offers the benefit of simplicity, since it

[4] Though there is a recent trend towards multi-modal inputs, we will focus on large **language** models in the strict sense here: The model's input and output are text sequences.

involves no model fine-tuning, thereby allowing us to keep a single generalist model.

Specifically, we focus on the key step of document verbalization, which raises several interesting questions: How well do LLMs perform at document comprehension tasks that involve challenging layout reasoning, even with no/little information on document geometry? How are LLMs influenced by the way we feed them document representations and particularly, can we alter the document representations in a way that allows a LLM to exploit document geometry to achieve the same performance as a multi-modal model?

We investigate these questions with experiments on several document understanding datasets including tasks from the DUE benchmark, SROIE, WebSRC, and proprietary Key Information Extraction (KIE) datasets (from real-world industry scenarios). We examine two LLMs, namely ChatGPT3.5[5] and the open-source LLM Solar[13]. Overall, we make the following contributions:

1. A novel rule-based approach that enriches the prompts of existing text-centric LLMs with spatial structure information from documents. The approach works across various kinds of documents and tasks and can be applied to various layouts without the need for fine-tuning.
2. A set of comprehensive experiments using both research and real-world document datasets as well as commercial and open-source models. We cover various document-specific tasks, different reading orders, and effects of noise being added to the OCR data.
3. Besides quantitative results, we also explore LLMs' limitations when it comes to interpreting document layout in-depth on particularly challenging cases, for which we have annotated a subset of SROIE. [6].

2 Related Work

LLMs: Language models built upon the attention-based transformer architecture [14] are probably among the currently most intensely studied models in AI. Due to the high growth they experienced, often involving several billion parameters, the capacity and reasoning capabilities of these models have rapidly progressed [12]. In addition to commercial providers such as OpenAI [15], a variety of open-source models such as Llama 2 [16] or Solar[13] are currently evolving. Two fundamental types of models are distinguished: (1) Encoders, which generate representations of input texts and use them to make decisions about texts. They are equipped with additional head layers that are be fine-tuned to the specific problem. (2) Decoders that generate text and can be instructed using prompts without additional training. Recently, the latter paradigm has emerged as the dominant approach, as the resulting LLMs can serve as generalist agents for ad-hoc problem solving, without fine-tuning to specific tasks. Instruction

[5] gpt-3.5-turbo-1106

[6] Our annotated SROIE-Challenge dataset is available for future research, see Section 4.1.

tuning is used as an additional training step to facilitate this: It aims to bridge the gap between the LLM's goal of next-word prediction and the user's goal of having the LLM follow human instructions. [17]. Accordingly, we focus on instruction-tuned decoder models in this work.

Multi-Modal Models: Many multi-modal models outsource OCR into preprocessing and operate on a combined input of document image and recognized text+geometry [5, 7, 18]: For example, the LayoutLM series, including the most recent version LayoutLMv3 [19, 20, 21], utilizes a BERT-type transformer encoder [22], which feeds on a concatenation of word embeddings and visual patch embeddings, and is trained with several masked language modeling (MLM) and word/patch alignment tasks. The model is applied to downstream tasks via fine-tuning specialied head models. Similarly, DocFormer [23] applies an early fusion of image and text signals and a pre-training with global text-image alignment. UDOP [24] follows a generative approach and reconstructs text layout by an encoder-decoder model.

Other models operate end-to-end, feeding only on the document image and addressing text understanding in the process: Donut [9] uses an encoder-decoder architecture, which is pretrained to recognize the document images' text on large-scale real-world (IIT-CDIP) and synthetic documents. Similarly, Dessurt [25] integrates OCR as part of its model. Many of the aforementioned papers include ablation studies demonstrating that models benefit from including geometry information in the input – when trained accordingly. In this work, we extend this question to instruction-tuned LLMs.

The work most similar to ours is LATIN-Prompt by Wang et al. [26], who have recently proposed a combination of a layout-aware document representation and a task-aware prompting, and have also investigated fine-tuning in the process. We extend on this work by (1) investigating multiple verbalization strategies, (2) thoroughly treating the prompt templates as a free, dataset-agnostic parameter to be optimized carefully and independently from the verbalization, and (3) exploring the limitations of LLMs' layout reasoning capabilities in more detail by inspecting challenge cases and evaluating the effect of layout and OCR inaccuracies.

3 Approach

Figure 1 shows an overview of our approach: Given a document, we extract its text and corresponding word geometries using off the shelf OCR solutions. The document is converted into a purely textual representation, using a step we refer to as *verbalization*. We propose different verbalization strategies to add geometric and layout information to the textual document representation (see Section 3.1). To study the robustness of the verbalization with respect to inaccuracies of OCR geometries we degrade the OCR before verbalization by either applying noise to word positions or emulating layout analysis errors. The verbalized document is then inserted into a prompt template together with task-specific directives, e.g.

questions to be answered (Section 3.3). The prepared prompt is then fed into an LLM and the response is parsed from the output.

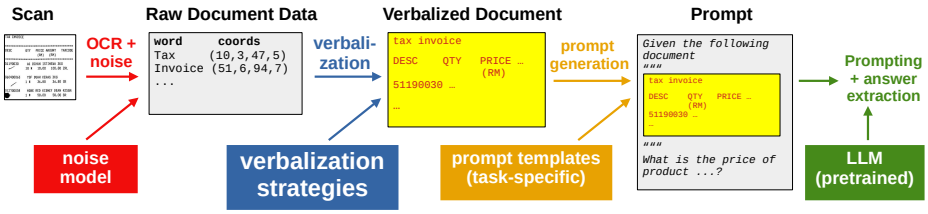

Fig. 1: Overview of our approach: Document OCR is converted into a text representation using different verbalization strategies (blue). Before verbalization, we optionally degrade the OCR by applying noise to the spatial position of OCR geometries (red). The resulting document text representation is then inserted into a task specific prompt (yellow) and fed into a LLM (green). Finally, the answers are extracted from the LLM output.

3.1 Verbalizers

We refer to verbalizers as strategies that create a textual document representation from an ordered collection of bounding boxes and the text associated with these boxes. This representation can serve as input to a text-based LLM. Further, each verbalizer offers a textual description of its output format to guide the LLM in interpreting the verbalizer's outputs. We outline multiple different verbalization strategies in the following. For each, we include the verbalization of an example word box with the text "TAX INVOICE" and coordinates $(x_{left}, y_{top}, x_{right}, y_{bottom}) = (100, 50, 321, 100)$ and center point $(x, y) = (211, 75)$:

1. **PlainText** Serves as a baseline by only adopting the text t without extra layout information. The text lines retrieved from the OCR are concatenated using newlines to form the document representation. When no line candidates are available, we concatenate words with spaces.
2. **BoundingBox** Uses both the bounding box coordinates and the text. The box geometries are encoded together with the text of each box using a custom format. Coordinates are rounded to whole numbers and are encoded as "left", "top", "right" and "bottom". Example:
`left:100 top:50 right:321 bottom:100 text:'TAX INVOICE'`
3. **BoundingBoxMarkup** Formats the bounding box coordinates in a XML style markup format followed by the text. Coordinates are rounded to whole numbers and are encoded as "left", "top", "right" and "bottom". Example: `<box left=100 top=50 right=321 bottom=100/>TAX INVOICE`

4. **CenterPoint** Formats both the bounding box center point coordinates in a XML style markup format followed by the text. Coordinates are rounded to whole numbers. Example:
   `<box x=211 y=75/>TAX INVOICE`
5. **SpatialFormat** Uses the geometries to restore the original document layout via insertion of spaces and newlines. To this end, the characters are placed on a grid such that their spatial location is similar to that on the document. Figure 2 shows a example output of the SpatialFormat verbalizer. At most 4 consecutive newlines are inserted. This approach is similar to LATIN-Prompt [26], see section 4.3 for a more detailed comparison.
6. **SpatialFormatY** Similar to SpatialFormat, but it only encodes spatial information on the vertical dimension, i.e. only newlines are inserted and no spaces are used for horizontal alignment. At most 4 consecutive newlines are inserted.
7. **PlainHTML** Serves as a control run for the WebSRC dataset, where a structured HTML representation of the document is available. Example:
   `...<h3 tid=‘‘3’’>TAX INVOICE</h3>...`

When verbalizing with SpatialFormat and SpatialFormatY, each page is verbalized individually. The resulting page verbalizations are then concatenated with an empty newline.

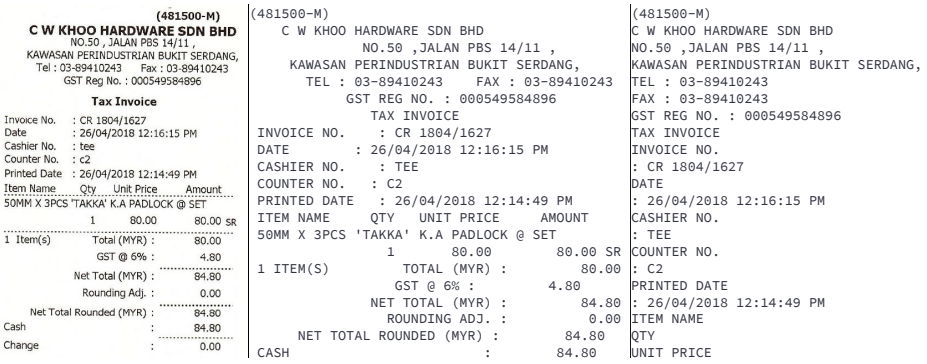

Fig. 2: Verbalization strategies on a random sample from the SROIE dataset: original (left), SpatialFormat (middle) and PlainText (right).

3.2 Noise Models

The OCR geometries generated by common OCR systems are subject to fluctuations and are rarely perfectly aligned with each other. To study the robustness of the verbalization strategies with respect to inaccuracies in the order and spatial relationship of layout elements, we optionally apply noise models to the OCR

output before feeding it into the verbalizers. Each noising model takes an ordered list of bounding boxes as input, where the initial order corresponds to the reading order of the underlying OCR engine.

1. `NONE`: Identity function. The coordinates and text of the OCR are not modified (serves as a control run).
2. `TRANSLATE`: Degenerates each bounding box b_i according to the formula $(x_0, y_0, x_2, y_2) \rightarrow (x_0+\Delta_i^x, y_0+\Delta_i^y, x_2+\Delta_i^x, y_2+\Delta_i^y)$, where $\Delta_i^x, \Delta_i^y \in [-20, 20]$ are uniformly sampled random numbers per box. Note that 20px is approximately the average character width in our data, such that two boxes can move up to 40px (or two letters) relative to each other.
3. `SHUFFLE`: Shuffles the list of bounding boxes randomly.
4. `NEAREST_NEIGHBOR`: Reorders a list of bounding boxes by selecting for each bounding box a successor box which is closer than `min_char_height` and `min_char_width` pixels. When there are no or multiple candidates, the successor box is selected under consideration of the original order of the boxes. The procedure emulates the *natural* reading order mode of Microsoft OCR[7] and tends to read tables column wise instead of row wise, as the spacing between consecutive rows is usually smaller than between consecutive columns.

3.3 Prompts

To prompt the LLM, we insert the verbalized document into a task-specific prompt template. As this prompt template influences quality just like the verbalization (order, wording and phrasing appear to matter), we aim to rigorously separate the effects of prompting from the effects of verbalization. To determine a suitable prompt structure, we subdivide each prompt into common building blocks and determine an optimal composition. Following known guidelines for prompt creation [27], we generate 10 different prompt structures and evaluate them on a small subset of our data. These structures differ in their ordering of the individual building blocks and are evaluated using a Question Answering (QA) task on the SROIE Challenge dataset (see Section 4.1).

Our prompts are divided into four components: `DOCUMENT` corresponds to the verbalized document representation. `TASK` encodes the task to solve. `FORMAT` describes the format used for verbalization. Finally, `OUTPUT` describes the expected output format using an example. We identified two patterns $\mathbf{A}, \mathbf{B}$ to work best: `DOCUMENT TASK OUTPUT` (pattern $\mathbf{A}$) and `DOCUMENT TASK FORMAT OUTPUT` (pattern $\mathbf{B}$). Based on these two structures, we create prompt templates for the tasks KIE, QA and NLI (natural language inference). For efficency reasons, we group multiple questions (QA), statements (NLI) or keys which are to be retrieved (KIE) into a single prompt. For QA and NLI samples that contain multiple questions to be answered, we enumerate those starting with 0. Figure 3 shows an example for QA prompt $\mathbf{B}$ with multiple questions.[8].

[7] This behaviour of *natural* reading order mode of Microsoft OCR has been shown empirically through our experiments.

[8] Please refer to Appendix A for a comprehensive overview of the prompt templates

```
$$$
<<<CONTENT>>>
$$$

From the above document, which is enclosed by "$$$", answer the
    following questions:
(0) which country had the most cyclists finish within the top 10?
(1) who was the first cyclist to finish?
(2) who came in first in the general standings?

In the document, the original layout was attempted to be restored
    via insertion of spaces and newlines.

The questions are numbered, e.g. "(0)".
Write the answers into a JSON dictionary and use the question
    numbers as keys and as datatype string.
Here is an example of the expected JSON format:
{
    "0": <ANSWER_TO_QUESTION_0>,
    "1": <ANSWER_TO_QUESTION_1>,
    ...
}
```

Fig. 3: Example for QA prompt **B** with three questions. Pattern **B** structure is: `DOCUMENT` (black), `TASK` (blue), `FORMAT` (orange) and `OUTPUT` (green).

3.4 Answer Extraction

Due to the probabilistic nature of LLMs as text generators, their outputs are not guaranteed to conform with the requested format. To ensure good readout of the answers, we process the output as follows: (1) We request a single JSON object which assigns the answers to the respective enumeration numbers (QA, NLI) or keys (KIE). (2) Given a single valid response object we parse the answers for the questions. (3) Given multiple valid response objects we choose the object with the most answers for the questions asked. (4) We use the enumeration number (QA, NLI) or the key (KIE) to extract a specific answer from the selected object. (5) If no valid JSON object is returned, we do not generate any answer. See Figure 3 for an example of the output format specification.

While the generation of JSON works reliably in most cases, LLMs will occasionally generate output that does not parse to valid JSON objects. Edge cases that we observed during development involve JSON objects which contain the correct value but a hallucinated key, e.g. `price_of_green_tea` instead of `answer`. Another common mistakes are nested objects, e.g. `{“price”: {“green_tea”: ...} }`. In these cases no answer is extracted.

4 Experiments

In the following experiments, we investigate whether suitable verbalization strategies can support LLMs with better layout reasoning and provide exemplary comparisons of open-source and commercial solutions. In most experiments, we measure the awareness of the LLM towards layout aspects via accuracy on document understanding tasks (which include research benchmarks and industry

datasets, see Section 4.1). To investigate layout awareness in depth, we also take a qualitative look at a subset of manually annotated challenge cases (see Section 4.3).

4.1 Datasets

DUE Benchmark We evaluate our approach using the DUE benchmark [1], specifically on the datasets *DocVQA*, *InfographicsVQA*, *TabFact* and *WikiTableQuestions* with the tasks VQA (DocVQA & InfographicsVQA), TableNLI (TabFact) and TableQA (WikiTableQuestions). We did not analyze the other datasets DeepForm, Kleister Charity and PWC, which are also part of the due benchmark, as these documents have a very high number of pages[9].

WebSRC WebSRC [28] is a collection of 360K question-answer pairs, which are collected from 60 different websites spanning 11 different domains. Besides the QA pairs the dataset also consists of web page segments, where each consists of a simplified version of the source HTML, a screenshot and a JSON file which contains additional spatial and layout information. Due to difficulties in retrieving text level bounding boxes from the JSON and HTML data, we manually perform OCR on the screenshots and use this data for further evaluation.[10] Only this dataset uses the PlainHTML verbalizer with the given HTML.

SROIE and SROIE Challenge SROIE [29] is a collection of 973 scanned receipts and the corresponding OCR results.[11] The task of the dataset is KIE with 4 keys to be extracted for each sample: company, date, address and total.

The original SROIE asks for the same 4 keys to be extracted for each sample. We argue that these keys in particular require no comprehensive understanding of the document's layout. For example, the company name is almost always the first thing written on the receipt, where date and address follow shortly after. To investigate LLM's understanding of layout more closely, we created a challenge set that queries the value of a specific table cell (*"How many of the item 'Green Tea' were purchased?"*) or directly reference the document's layout (*"Which entity is written above the card expiry date?"*). To do so, we manually annotated 101 samples from the train split of the SROIE dataset to create a challenging QA dataset[12]. We categorize the questions into *quantity*, *currency* and *string*, where the latter refers to any other question that corresponds to neither of the first two types.

Proprietary KIE Datasets We further evaluate KIE performance on two proprietary KIE datasets from Insiders Technologies, which both contain particularly diverse and challenging examples from real world business correspondence:

[9] A limitation when working with long documents is the context length of LLMs. While solutions to this exist, such verbose documents are not part of our scope.

[10] This OCR data has been contributed to the authors of WebSRC and is also made publicly available at https://github.com/46692/WebSRC_OCR

[11] We use the revised version of the dataset from https://www.kaggle.com/datasets/urbikn/sroie-datasetv2

[12] Made publicly available at https://github.com/46692/SROIEChallenge

ITForms is a collection of 100 multipage form documents in German language, with 9 keys each. It includes, among others, forms for applying for insurance benefits, registering vehicles and bank forms, e.g. opening a depot. **ITInvoices** is a collection of 104 invoice documents in German language, which are predominantly single page and contain 21 keys each. It includes both business invoices as well as scanned receipts.

4.2 Setup

LLMs We evaluate our approach with two LLMs: ChatGPT and Solar[13]. For the evaluation of ChatGPT we use `gpt-3.5-turbo-1106`[13] in JSON mode[30], with a temperature of 0, and enter each prompt in the role of `user`. We further evaluate the 8 bit quantized version[14] of the recent open-source LLM Solar 70b on SROIE and SROIE Challenge.[15] For Solar each prompt is also entered in the role of `user`.[16] Unless stated otherwise, experiments use the ChatGPT model and prompt template **A**, i.e. the template without verbalizer format description.

OCR Each dataset in the DUE benchmark comes with a selection of pre-applied OCR engines, where we use `microsoft_cv` and `tesseract` as a fallback in case the former is not available, which is only the case for TabFact. The OCR results in these datasets contain information about the page and line index, which is used to join all word bounding boxes on the same page with the same line index together. Microsoft Computer Vision OCR is used for WebSRC, ITForms and ITInvoices. We contribute the OCR for WebSRC train and test splits. For SROIE and SROIE Challenge we use the OCR results delivered with the dataset.

Metrics For evaluation of the DUE datasets we use the official evaluation framework[17] with the metrics given in [1]: ANLS for DocVQA and InfographicsVQA and accuracy for TabFact and WikiTableQuestions. WebSRC is evaluated according to the procedure in the GitHub repository[18] and the scores are given as EM (exact match) and F1 (harmonic mean of recall and precision). For SROIE, SROIE Challenge, ITForms and ITInvoices we create a type aware accuracy measure: Each response is assigned to one of four types based on the expected response, which describe how it is compared to the ground truth (the procedures described are applied to both the ground truth and the response extracted from the LLM output): For `string` values a case-insensitive comparison is made. `date` values are parsed via the Python dateparser library[19] and then

[13] The model was used in the period of November 2023 to February 2024.
[14] https://huggingface.co/upstage/SOLAR-0-70b-8bit
[15] The evaluation of other datasets had to be omitted due to time constraints.
[16] As stated on the Hugging Face model card, Solar expects the role being given in the prompt. Therefore we used the following wrapper `### User:PROMPT\n\n\n### Assistant:` where `PROMPT` is replaced with the prompt prepared by our pipeline and `\n` symbolizes an empty line.
[17] https://github.com/due-benchmark/evaluator
[18] https://github.com/X-LANCE/WebSRC-Baseline
[19] https://github.com/scrapinghub/dateparser

compared for equality. `currency` values are sanitized via a regular expression[20] (RegEx), replacement of commas with dots and then compared for equality. `quantity` values are sanitized via a RegEx[21] and then compared for equality. The proposed accuracy measure defaults to EM for `string` and also for `currency` and `quantity` after units are neglected, i.e. no rounding is performed for the latter two. In case a value cannot be parsed to its specified type, an empty answer is returned.

4.3 Results

See Section B in the appendix for a comparison of the token overhead added by each verbalization strategy. See Section C in the appendix for an analysis of the effects that the verbalizer format description has on the different prompt templates A and B.

Dataset Results We report the results on the various datasets with the metrics laid out in section 4.2: type aware accuracy for SROIE, SROIE Challenge, ITForms, ITInvoices; ANLS for DocVQA, InfographicsVQA; accuracy for TabFact, WikiTableQuestions; F1 and EM for WebSRC. The results in tables 1 and 2 show, that our approach can compete with state-of-the-art models. Specifically, the results of the DUE benchmark in table 1 demonstrate that the introduction of layout information to the prompt proves beneficial. Our approach achieves state-of-the-art performance on InfographicsVQA and WikiTableQuestions.[22] Throughout the benchmark, SpatialFormat proves to be the best verbalization strategy on average, with a peak gain of 15% (from 47.7% to 54.9%) on InfoVQA.[23] In comparison to LATIN-Prompt [26], our SpatialFormat approach results in slightly different formatting (i.e. less inserted whitespace and also incorporates newlines) but is overall almost identical, thereby confirming their results. While LATIN uses dataset specific prompt templates, we use task specific prompt templates.

Table 2 shows that we achieve competitive results for some of the other datasets. Specifically, our approach achieves the 3rd best F1 score on WebSRC with the SpatialFormat verbalization.[24] The PlainHTML baselines further shows promising results for HTML formatted document representations, achieving best performance out of all verbalizations. However, this verbalization strategy is not viable for real world documents, as these would have to exist as HTML documents in the first place or would introduce a separate layout processing model into the pipeline, eliminating the need for our approach. Results on SROIE show that StructTexT significantly outperforms our approach, demonstrating the superiority of multi-modal models on the dataset. Notably, PlainText performs best among the verbalizations, which could be explained by the

[20] `\d+(?:(\.|,)\d1,2)?`
[21] `(?:[ a-zA-Z]*)(\d+)(?:[ a-zA-Z]*)`
[22] State as of 10th February 2024 according to https://duebenchmark.com
[23] Note that questions in InfographicsVQA focus on text rather than on graphics.
[24] State as of 10th February 2024 according to [31]

simplicity of the dataset paired with the given OCR line segments, resulting in simple key-value pairs in most cases. Comparison with the other datasets is difficult: For our custom SROIE Challenge subset no comparisons exist of course. While ITForms and ITInvoices are proprietary datasets that do not allow direct comparison with other approaches, they give us an insight into how the models operate on real world business documents. These data sets are characterized by the fact that not all keys have to generate a value. Information is often missing on real documents and not every key can be assigned to a value. The model must therefore have sufficient ability to reject a value, i.e. it should only output a value if it can be found on the document. Our results show that the LLM-based approaches perform significantly worse on these datasets. We observed that in most cases the LLMs produce outputs and rarely provide an empty response, which lowers their overall score in the evaluation. We believe that this problem can be reduced by clearer instructions in the prompt.

Table 1: Comparison with other models published on the DUE-Benchmark. Underlines denote the best verbalization strategy in the dataset. It shows that our approach achieves competitive results and even state-of-the-art results on the two datasets InfographicsVQA and WikiTableQuestions, with an improvement of 15% compared to the baseline for the former. Further, it is shown that SpatialFormat is the best verbalization strategy among the ones tested.

Model	Modality	Question Answering		Table QA/NLI		Avg.
		DocVQA	InfoVQA	WTQ	TabFact	
$\text{BERT}_{\text{LARGE}}$ [22]	T	67.5	-	-	-	-
Donut [9]	V	72.1	-	-	-	-
T5_{LARGE}+2D+U [32]	T+L	81.0	46.1	43.3	78.6	62.3
$\text{LayoutLMv2}_{\text{LARGE}}$ + QG [20]	T+L+V	**86.7**	-	-	-	-
$\text{LayoutLMv3}_{\text{LARGE}}$ [21]	T+L+V	83.4	45.1	45.7	78.1	63.1
UDOP [6]	T+L+V	84.7	47.4	47.2	**78.9**	**64.6**
LATIN-Prompt (Claude) [26]	T+L	82.6	54.5	-	-	-
Ours PlainText	T	76.3	47.7	45.1	68.4	59.4
Ours SpatialFormat	T+L	<u>79.8</u>	<u>**54.9**</u>	<u>**47.7**</u>	70.1	<u>63.1</u>
Ours SpatialFormatY	T+L	76.3	49.6	45.5	<u>70.3</u>	60.4
Ours BoundingBox	T+L	74.8	46.4	35.0	68.5	56.2
Ours BoundingBoxMarkup	T+L	74.6	45.8	36.2	68.6	56.3
Ours CenterPoint	T+L	75.1	47.4	38.2	67.8	57.1

Comparison of ChatGPT to Solar We compare the performance of our approach when applied to two different LLMs, specifically ChatGPT 3.5 and Solar70B8Bit. The results in table 3 show that open-source LLMs provide a viable alternative to commercial solutions for document comprehension using

Table 2: Evaluation results for SROIE, ITForms, ITInvoices, WebSRC and SROIEChallenge. Underlines denote the best verbalization strategy for the dataset. WebSRC results of other models are taken from the official leaderboard and show that the performance our approach is close to that of the third-placed model. Proprietary KIE Model refers to an internal model of Insiders Technologies, which is a multi-modal LLM free approach. ITForms and ITInvoices contain samples for which not all keys have a value on the documents. While this works to some extent, it is not properly supported using our current prompt. *For WebSRC left score is EM and right score is F1.

Model	Modality	KIE			Question Answering	
		SROIE	ITForms	ITInvoices	WebSRC*	SROIEChallenge
SageGPT-small-v0.2 [31]	?	-	-	-	**89.1 / 92.2**	-
DocPrompt (ErnieLayout-Large) [33]	T+L+V	-	-	-	77.4 / 85.0	-
TIE (MarkupLM-Large) [34]	T+L	-	-	-	76.3 / 80.5	-
StructTexT [35]	T+L+V	**98.7**	-	-	-	
Proprietary KIE Model	T+L	91.7	**86.2**	**90.1**	-	-
Ours PlainText	T	79.9	68.4	54.5	72.9 / 80.5	81.2
Ours SpatialFormat	T+L	77.0	73.9	54.2	74.2 / 80.7	**86.1**
Ours SpatialFormatY	T+L	79.0	69.0	54.6	72.4 / 80.3	81.2
Ours BoundingBox	T+L	75.4	64.2	54.1	68.3 / 76.6	72.3
Ours BoundingBoxMarkup	T+L	74.3	65.6	53.9	68.1 / 75.9	71.3
Ours CenterPoint	T+L	73.3	65.1	51.8	68.9 / 76.9	74.3
Ours PlainHTML	T+L	-	-	-	80.0 / 84.1	-

our approach: The performance of both LLMs is similar on SROIE, with Solar performing slightly better. On SROIE Challenge, ChatGPT has a lead of 4.8 pp. on average. Further, it is shown that Solar is apparently able to make better usage of the layout information delivered by BoundingBox, BoundingBoxMarkup and CenterPoint verbalizers compared to ChatGPT. While it is unclear whether SROIE is part of ChatGPT's training data, we checked the training data of Solar[25] and could find no SROIE data. However, we can assure that neither of both models has seen the questions of SROIE Challenge during training.

Noise Model Analysis We evaluate the robustness of our verbalization strategies against noise and layout misinterpretations introduced to the document data. We simulate this by applying the noise models `TRANSLATE`, `SHUFFLE` and `NEAREST_NEIGHBOR` (see Section 3.2). For each noise model, the average of the scores achieved with each verbalization strategy across all datasets is determined. [26] The results presented in figure 4 show that SpatialFormat and SpatialFormatY are the least affected by the noise models. Further, it shows that PlainText is very susceptible to wrong layout interpretation by the OCR system.

[25] As given under https://huggingface.co/upstage/SOLAR-0-70b-8bit

[26] In line with the DUE benchmark, we resort to an arithmetic mean of different metrics. [1] For WebSRC, where two metrics are reported, we use the F1 score.

Table 3: Evaluation results for the comparison of Solar and ChatGPT 3.5. Underlines denote the best verbalization strategy for the LLM in the dataset. It shows that open-source LLMs provide a viable alternative to commercial solutions for document comprehension using our approach: Solar performs slightly better than ChatGPT on SROIE, while ChatGPT has an advantage of 4.8 pp. on our custom SROIE Challenge set.

Verbalizer	Modality	SROIE		SROIE Challenge	
		ChatGPT	Solar	ChatGPT	Solar
PlainText	T	<u>79.9</u>	76.6	81.2	72.3
SpatialFormat	T+L	77.0	76.7	<u>86.1</u>	<u>81.2</u>
SpatialFormatY	T+L	79.0	77.3	81.2	79.2
BoundingBox	T+L	75.4	76.2	72.3	66.3
BoundingBoxMarkup	T+L	74.3	<u>77.7</u>	71.3	66.3
CenterPoint	T+L	73.3	76.9	74.3	72.3
Avg.		76.5	**76.9**	**77.7**	72.9

Qualitative Analysis: SROIE Challenge We explore the impact of document layout on LLMs in-depth on our SROIE Challenge dataset, which features demanding questions on specific table cells and the relative position of items. See Section D in the appendix for examples of these challenge cases. We found the LLM to work surprisingly well, answering 59 of 101 questions with all verbalizers and 82 with the PlainText verbalizer. The failures on the remaining 19 samples were related to layout misinterpretations that follow directly from the limited plain-text verbalization: (i) column-wise order of OCR output instead of a row-wise (ii) delayed table cells, which were placed after all other cells at the end of a table. (iii) overly complex samples (e.g. tables spanning over 12 rows and 6 columns) (iv) empty cells, which lead the LLM to wrong conclusions based on the ordering of the bounding boxes, and (v) neighboring cells merged to a single bounding box. Especially combinations of these factors provided challenging cases. Overall, the challenge cases were more reliably solved by the SpatialFormat strategy (87 out of 101 correct). This indicates – on a limited number of manually inspected samples – a better resiliency of the SpatialFormat verbalization with respect to OCR layout misinterpretations.

5 Conclusion

We have investigated techniques for adding layout information to prompts for instruction-tuned LLMs to enhance document understanding performance. This approach only requires pre-processing of document text and the prompt without the need for extra fine-tuning. We achieve higher scores compared to layout-unaware document representations on 7 out of 9 datasets across different document tasks, reaching state-of-the-art results on two datasets and often times

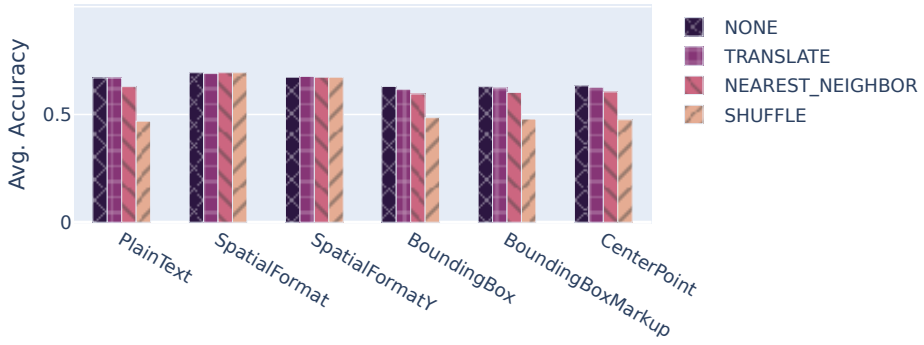

Fig. 4: Comparison of the noise models for each verbalizer. Values shown are scores averaged over all datasets. It shows, that PlainText's performance diminishes when the layout is misinterpreted. Further is shown that SpatialFormat and SpatialFormatY verbalizers are the least prone to noise introduced to the OCR data. Note that they are not affected by changes to the bounding box ordering, as they operate only using the bounding box coordinates.

yielding results competitive with those of specially trained multi-modal models. We have shown that our approach works for both commercial as well as open-source LLMs. A potential threat to validity is that our datasets may have been part of the training data for ChatGPT and Solar.[27] Our results indicate, however, that the improvements of our approach on the non-public datasets (ITForms, ITInvoices) and the specifically annotated one (SROIE-Challenge), are in line with the findings on public datasets. The proposed method is particularly suited for structured documents that make heavy use of spatial alignments and blanks.

For future research, an interesting subject are recent multi-modal instruction-tuned LLMs with additional visual input such as GPT-4 [37]. Due to both time constraints and the higher cost associated with these models, we have focused on text-only representations in this work. Also, evaluation of the presented approach on languages with different reading orders (e.g. Arabic) would be of interest. Extending SpatialFormat to handle documents with different reading orientations and text overlaps would further strengthen the method. Another obvious direction is the evaluation of a larger number of LLMs (including a comparison of error cases), which had to be omitted due to time constraints. Of particular interest is the evaluation of the influence of the number of parameters on the performance of the proposed method. We also recon that more work is needed when scaling our solution to multi-page reasoning problems, especially when the number of pages becomes larger.

[27] Solar [13] uses weights of Mistral 7B [36], for which the training dataset is not publicly available.

References

[1] Łukasz Borchmann et al. "DUE: End-to-End Document Understanding Benchmark". In: *NeurIPS Datasets and Benchmarks*. 2021. URL: https://api.semanticscholar.org/CorpusID:244906279.

[2] Minghao Li et al. "Tablebank: Table benchmark for image-based table detection and recognition". In: *Proceedings of the Twelfth Language Resources and Evaluation Conference*. 2020, pp. 1918–1925.

[3] Minghao Li et al. "DocBank: A benchmark dataset for document layout analysis". In: *arXiv preprint arXiv:2006.01038* (2020).

[4] Yiheng Xu et al. "XFUND: A Benchmark Dataset for Multilingual Visually Rich Form Understanding". In: *Findings of the Association for Computational Linguistics: ACL 2022*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 3214–3224. URL: https://aclanthology.org/2022.findings-acl.253.

[5] Haoyu Cao et al. "GMN: Generative Multi-modal Network for Practical Document Information Extraction". In: *arXiv preprint arXiv:2207.04713* (2022).

[6] Zineng Tang et al. "Unifying vision, text, and layout for universal document processing". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 19254–19264.

[7] Chuwei Luo et al. "GeoLayoutLM: Geometric Pre-training for Visual Information Extraction". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 7092–7101.

[8] Dongsheng Wang et al. "DocLLM: A layout-aware generative language model for multimodal document understanding". In: *arXiv preprint arXiv:2401.00908* (2023).

[9] Geewook Kim et al. "OCR-Free Document Understanding Transformer". In: *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVIII*. Tel Aviv, Israel: Springer-Verlag, 2022, pp. 498–517. URL: https://doi.org/10.1007/978-3-031-19815-1_29.

[10] Tengchao Lv et al. "Kosmos-2.5: A multimodal literate model". In: *arXiv preprint arXiv:2309.11419* (2023).

[11] Yi-Hsueh Liu et al. "Summary of ChatGPT/GPT-4 Research and Perspective Towards the Future of Large Language Models". In: *ArXiv* abs/2304.01852 (2023). URL: https://api.semanticscholar.org/CorpusID:263893278.

[12] Jason Wei et al. "Emergent Abilities of Large Language Models". In: *Trans. Mach. Learn. Res.* 2022 (2022). URL: https://openreview.net/forum?id=yzkSU5zdwD.

[13] Dahyun Kim et al. *SOLAR 10.7B: Scaling Large Language Models with Simple yet Effective Depth Up-Scaling*. 2023. arXiv: 2312.15166 [cs.CL].

[14] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[15] OpenAI. "GPT-4 Technical Report". In: *ArXiv* abs/2303.08774 (2023). URL: https://arxiv.org/abs/2303.08774.

[16] Hugo Touvron et al. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. July 2023.

[17] Shengyu Zhang et al. *Instruction Tuning for Large Language Models: A Survey*. 2023. arXiv: 2308.10792 [cs.CL].

[18] Hao Feng et al. "Unidoc: A universal large multimodal model for simultaneous text detection, recognition, spotting and understanding". In: *arXiv preprint arXiv:2308.11592* (2023).

[19] Yiheng Xu et al. "LayoutLM: Pre-training of Text and Layout for Document Image Understanding". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD '20. Virtual Event, CA, USA: Association for Computing Machinery, 2020, pp. 1192–1200. URL: https://doi.org/10.1145/3394486.3403172.

[20] Yang Xu et al. "LayoutLMv2: Multi-modal Pre-training for Visually-rich Document Understanding". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Ed. by Chengqing Zong et al. Online: Association for Computational Linguistics, Aug. 2021, pp. 2579–2591. URL: https://aclanthology.org/2021.acl-long.201.

[21] Yupan Huang et al. "LayoutLMv3: Pre-training for Document AI with Unified Text and Image Masking". In: *Proceedings of the 30th ACM International Conference on Multimedia*. MM '22. Lisboa, Portugal: Association for Computing Machinery, 2022, pp. 4083–4091. URL: https://doi.org/10.1145/3503161.3548112.

[22] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. URL: https://aclanthology.org/N19-1423.

[23] Srikar Appalaraju et al. "DocFormer: End-to-End Transformer for Document Understanding". In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)* (2021), pp. 973–983. URL: https://api.semanticscholar.org/CorpusID:235592814.

[24] Zineng Tang et al. "Unifying Vision, Text, and Layout for Universal Document Processing". In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022), pp. 19254–19264. URL: https://api.semanticscholar.org/CorpusID:254275326.

[25] Brian Davis et al. "End-to-End Document Recognition and Understanding with Dessurt". In: *Computer Vision – ECCV 2022 Workshops: Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part IV*. Tel Aviv, Israel: Springer-Verlag, 2023, pp. 280–296. URL: https://doi.org/10.1007/978-3-031-25069-9_19.

[26] Wenjin Wang et al. *Layout and Task Aware Instruction Prompt for Zero-shot Document Image Question Answering*. 2023. arXiv: 2306.00526 [cs.CL].
[27] *OpenAI Docs Prompt Engineering*. 2024. URL: https://platform.openai.com/docs/guides/prompt-engineering/six-strategies-for-getting-better-results (visited on 01/27/2024).
[28] Xingyu Chen et al. "WebSRC: A Dataset for Web-Based Structural Reading Comprehension". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 4173–4185. URL: https://aclanthology.org/2021.emnlp-main.343.
[29] Zheng Huang et al. "ICDAR2019 Competition on Scanned Receipt OCR and Information Extraction". In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. 2019, pp. 1516–1520.
[30] *OpenAI Docs JSON Mode*. 2024. URL: https://platform.openai.com/docs/guides/text-generation/json-mode (visited on 01/27/2024).
[31] URL: https://x-lance.github.io/WebSRC/.
[32] Rafał Powalski et al. "Going full-tilt boogie on document understanding with text-image-layout transformer". In: *Document Analysis and Recognition–ICDAR 2021: 16th International Conference, Lausanne, Switzerland, September 5–10, 2021, Proceedings, Part II 16*. Springer. 2021, pp. 732–747.
[33] Sijin Wu et al. "DocPrompt: Large-scale continue pretrain for zero-shot and few-shot document question answering". In: *arXiv preprint arXiv:2308.10959* (2023).
[34] Junlong Li et al. "MarkupLM: Pre-training of text and markup language for visually-rich document understanding". In: *arXiv preprint arXiv:2110.08518* (2021).
[35] Yulin Li et al. "Structext: Structured text understanding with multi-modal transformers". In: *Proceedings of the 29th ACM International Conference on Multimedia*. 2021, pp. 1912–1920.
[36] Albert Q. Jiang et al. *Mistral 7B*. 2023. arXiv: 2310.06825 [cs.CL].
[37] URL: https://openai.com/blog/chatgpt-can-now-see-hear-and-speak.