DocEditAgent: Document Structure Editing Via Multimodal LLM Grounding

Anonymous ACL submission

Abstract

Document structure editing involves manipulating localized textual, visual, and layout components in document images based on the user's requests. Past works have shown that multimodal grounding of user requests in the document image and identifying the accurate structural components and their associated attributes remain key challenges for this task. To address these, we introduce the DocEditAgent, a novel framework that performs end-to-end document editing by leveraging Large Multimodal Models (LMMs). It consists of three novel components – (1) Doc2Command to simultaneously localize edit regions of interest (RoI) and disambiguate user edit requests into edit commands. (2) LLM-based Command Reformulation prompting to tailor edit commands originally intended for specialized software into edit instructions suitable for generalist LMMs. (3) Moreover, DocEditAgent processes these outputs via Large Multimodal Models like GPT-4V and Gemini, to parse the document layout, execute edits on grounded Region of Interest (RoI), and generate the edited document image. Extensive experiments on the DocEdit dataset show that DocEditAgent significantly outperforms strong baselines on edit command generation (2-33%), RoI bounding box detection (12-31%), and overall document editing (1-12%) tasks.

1 Introduction

Digital documents are widely used for communication, information dissemination, and business productivity. Language-guided Document Editing entails modifying the textual, visual, and structural components of a document in response to a user's open-ended requests related to spatial alignment, component placement, regional grouping, replacement, resizing, splitting, merging, and applying special effects (Mathur et al., 2023a; Kudashkina et al., 2020). Document editing is inherently a generative task as it involves the creation of a new edited output from an existing document.

Mathur et al. (2023a) highlights three key challenges in the end-to-end document editing task – (1) multimodal grounding of ambiguous user requests in the document image, (2) identifying the precise components and their corresponding attributes to be edited, and (3) generating faithful edits without distorting the semantic or spatial coherence of the original document. By interpreting the visual-semantic cues from user requests, multimodal grounding can bridge the gap between natural language instructions and the spatial intricacies of the document's content. Sophisticated edit commands, like those found in the DocEdit dataset (Mathur et al., 2023a), are usually ambiguous in nature and tailored for use in software-specific applications. Disambiguation of such edit commands can help to serve as refined editing instructions for generalist generation models. We hypothesize that directly editing the parsed HTML/XML document structure can overcome the limitations of pixel-level image generation.

Prior works like DocEditor Mathur et al. (2023a) performed edit commands generation for language-guided document editing but was limited to software-specific applications. Generative methods such as diffusion models have shown promise in the visual domain but pose challenges in recreating complex textual and visual elements while preserving the structural information of documents (Yang et al., 2023b; He et al., 2023). Unlike natural images, documents contain a combination of text, images, formatting, and layout intricacies (Mathur et al., 2023b) that necessitate a more nuanced approach to generative editing. Recently, Large Multimodal Models (LMMs) like GPT-4V (OpenAI, 2023) and Gemini (Team et al., 2023) have demonstrated remarkable capabilities in document understanding, object localization, dense captioning, and code synthesis. Prior work has also explored LLM

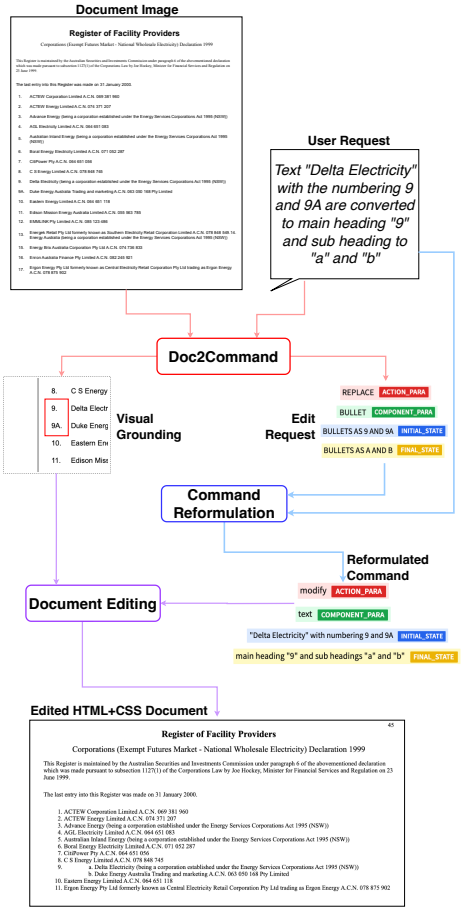


Figure 1: `DocEditAgent` framework performs multimodal grounding and edit command generation via Doc2Command, utilizes LLM-based Command Reformulation prompting to refine the command into LMM instruction format (< ACTION >< Component >, < Initial State >, < Final State >), and employs LMMs to edit the HTML structure using multimodal (edit instruction and grounded RoI) prompt.

program synthesis to compose vision-and-language queries into code subroutines (Gao et al., 2022; Sur'is et al., 2023; Feng et al., 2023; Huang et al., 2023). Our work aims to solve end-to-end editing of HTML representation of documents by leveraging the emergent capabilities of LMMs to infer the semantic context of edit requests, visually reference them to the region of interest in the document image, determine the spatial elements to be modified, and generate the final document.

Main Results: We present `DocEditAgent` (Fig.1) – an LMM-based end-to-end document editing framework. Given a user request on a document, it utilizes a novel Doc2Command module to ground the edit location in the document image and generate edit commands. Doc2Command is a Transformer-based image encoder-text decoder-mask transformer model that is jointly trained to perform masked semantic segmentation and ground edit regions of interest (RoI) for disambiguating user edit requests into modularized commands. Doc2Command starts with visually integrating the edit request with the document image, processing them as a unified visual modality through a vision encoder-text decoder backbone to generate the command text. It redefines bounding box detection as a segmentation task by incorporating a mask-attention transformer over the image encoder. Further, we propose Command Reformulation prompting to customize the edit commands into an LMM-specific editing instruction by leveraging the zero-shot in-context learning ability of LLMs. Lastly, `DocEditAgent` leverages LMMs such as GPT-4V and Gemini to edit the HTML structure of the document using a multimodal prompt formed by combining the edit instruction and grounded RoI. We design two new metrics - CSS IoU, and DOM Tree Edit Distance to evaluate the final edited documents for presentation quality and structural similarity with the ground truth. Experiments on the DocEdit dataset reveal that \texttt{DocEditAgent} significantly outperforms strong baselines in edit command generation (by 2-33%), RoI bounding box detection (by 12-31%), and overall document editing tasks (by 1-12%). Our **main contributions** are:

- We propose **Command Reformulation** to resolve ambiguity by using Large Language Models (LLMs) to translate the user's linguistic intent into a specific visual editing prompt for LMMs.
- We introduce **Doc2Command**, a novel model for grounding edit requests that employs a transformer-based image encoder and text decoder architecture. It generates precise commands for document editing and semantically anchors editing regions through masked semantic segmentation in a multitask framework.
- We present **DocEditAgent**, an LMM-based framework for document editing. It interprets user requests to perform localized editing tasks conversationally. DocEditAgent utilizes Command Reformulation to convert user intent into appropriate LMM prompts and incorporates multimodal grounding via our proposed Doc2Command module.

- Additionally, we define two new metrics - CSS IoU and DOM Tree Edit Distance - to assess LMM-generated documents for presentation quality and structural fidelity compared to ground truth.

2 Related Work

Past works in the domain of language-guided image editing have predominantly centered on natural image datasets (Shi et al., 2020; Lin et al., 2020), overlooking the distinctive characteristics of documents, which typically exhibit text-rich content alongside a diverse array of structured elements arranged in various layouts. These datasets often lack representations of localized edits and indirect edit references, crucial facets for effective document editing. Notably, contemporary GAN-based (Li et al., 2020; Jiang et al., 2021a,b; Cheng et al., 2020; Ling et al., 2021) and diffusion methods (Joseph et al., 2024; Kawar et al., 2023; Tumanyan et al., 2023; Brooks et al., 2023; Nichol et al., 2021) have gained traction for natural image manipulation tasks due to their capacity for end-to-end pixel-level image synthesis. However, their applicability to digital documents, characterized by rich textual content and complex layouts, remains limited. These techniques are ill-equipped to grasp the spatial and semantic intricacies inherent in embedded textual components within documents. Consequently, prior endeavors in language-guided document editing have primarily pivoted towards multimodal grounding of edit requests through textual and visual cues into actionable commands and visual localization (Mathur et al., 2023a). Despite these efforts, the absence of efficient generative frameworks tailored for document image editing remains a significant challenge in this domain.

3 `DocEditAgent` Methodology

`DocEditAgent` (Fig. 1) comprises of the following steps to ensure effective edit operation: (a) multimodal grounding and edit command generation via the Doc2Command, (b) Command Reformulation prompting to transform the edit command into LMM-specific prompt instruction, (c) prompting LMMs like GPT-4V and Gemini to facilitate nuanced and localized editing of the document's HTML representation.

3.1 Doc2Command

Editing documents based on user requests requires converting open-vocabulary user requests into precise actions and grounding the region of interest in the document image. Edit command generation involves semantically mapping the ambiguous natural language user requests to specific editing actions, components, and associated attributes to ensure that the intended modifications are accurately interpreted and executed. Multimodal grounding is essential to recognize the specific textual or visual document elements referenced by the user. Doc2Command is a multi-task, multimodal Transformer-based model aimed at jointly achieving both these objectives of region of interest segmentation and command generation.

Modeling Doc2Command: Doc2Command uses a pre-trained Vision Transformer (Dosovitskiy et al., 2021) (ViT) image encoder borrowed from Pix2Struct(Lee et al., 2023) which has been pre-trained with a text decoder for screenshot parsing via masked document image modeling objective. The patch embeddings generated by the encoder serve as input to the pre-trained Pix2Struct decoder and the mask transformer.

Edit Command generation: We strategically render the input text request as a text box element on the top of the document image. This approach allows for a more flexible integration of linguistic and visual inputs that can be processed jointly by the image encoder. Instead of scaling the input image to a pre-defined resolution, we adjust the scaling factor to maximize the number of fixed-size patches that can fit the image encoder's sequence length. This makes the model more robust against extreme aspect ratios of document images. Each patch is flattened to obtain a vector of pixels and then fed into the image encoder to generate patch encoding. The patch embeddings generated by the encoder serve as input to the text decoder, which auto-regressively generates a sequence of tokens representing the command text specified as: *ACTION(<Component>, <Initial State>, <Final State>)*, containing the action, its associated components, attributes, initial and final states. More details in Sec. A.6.

Multimodal Grounding: We approach the detection of bounding boxes through the lens of a semantic segmentation task. Given the bounding boxes for the region of interest and the rendered user request, we create ground truth segmentation maps

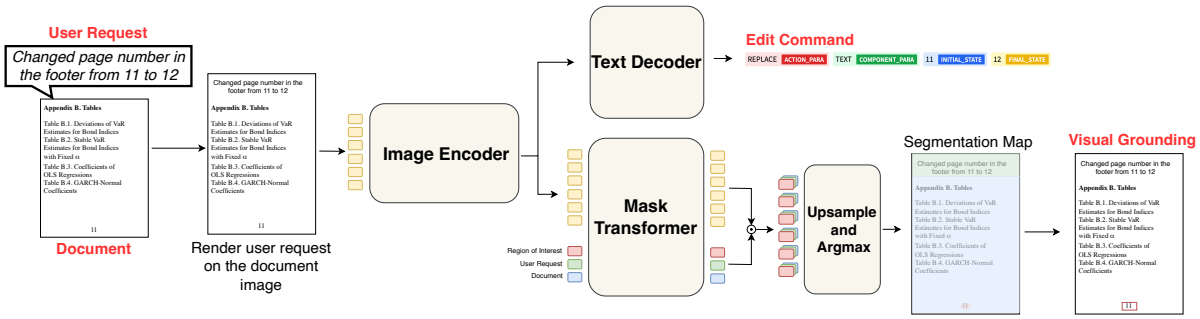


Figure 2: Doc2Command: Given a document image and a user request, the user request is rendered onto the document, and passed as a singular visual modality to an image encoder. The image encoder feeds into a text decoder and a mask transformer to generate the command text and segmentation maps, respectively.

with three classes: (1) the Region of Interest, (2) the rendered user request text, and (3) the remaining document. We utilize a DETR-style transformer (Carion et al., 2020) for masked attention modeling. A set of K learnable class embeddings ($K = 3$ for our model) is initialized randomly and assigned to a single semantic class. It is used to generate the class mask. The mask-transformer processes the class embeddings jointly with patch encoding and generates K masks by computing the scalar product between L2-normalized patch embeddings with class embeddings output by the decoder. The set of class masks is reshaped into a 2D mask and bilinearly upsampled to the image size to obtain a feature map, followed by a softmax and layer normalization to obtain pixel-wise class scores, forming the final masked segmentation maps that are softly exclusive to each other. At inference, the segmented area is converted into a bounding box by considering points within a 95% radius of the centroid of the mask. The contours of the largest contiguous object are then used to determine the coordinates of the bounding box, which is denoted by (x, y, h, w). Here, (x, y) is the top-left coordinate of the bounding box, h and w are height and width, respectively. More details in Sec. A.7.

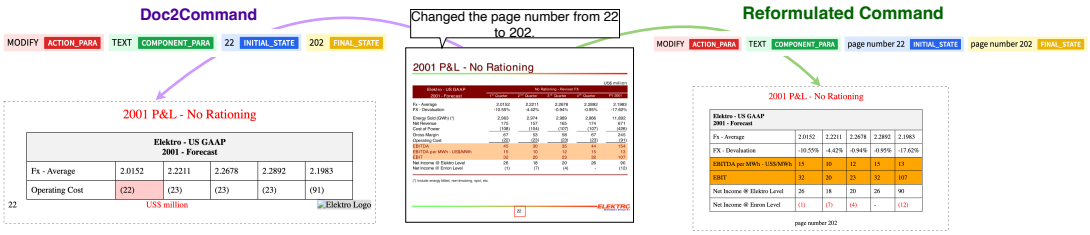
Training Doc2Command: The text decoder is fine-tuned to generate the command text, while the mask transformer is fine-tuned for segmentation. The multitask setup employs a combined weighted loss given by $\mathcal{L}_{\text{total}} = \lambda_{\text{text}} \cdot \mathcal{L}_{\text{text}} + \lambda_{\text{seg}} \cdot \mathcal{L}_{\text{seg}}$. The segmentation loss $\mathcal{L}_{\text{seg}}$ is itself a sum of focal loss (Lin et al., 2017) and dice loss (Sudre et al., 2017).

3.2 Command Reformulation Prompting

Doc2Command is trained on the command generation task from DocEdit dataset (Mathur et al., 2023a), which is geared towards generating software-specific commands. Consequently, the generated edit commands are sub-optimal to be used as editing instructions for generalist LMMs (see examples in Fig. 5-12). Additionally, the generated commands may underspecify the actions, components, and associated attributes needed to faithfully produce the final edit due to ambiguities in the user request. Hence, there is a need to reformulate the generated edit commands to perfectly align with the requisite format of the prompt instructions expected by generalist multimodal generation models like GPT-4V and Gemini. We address this limitation by introducing Command Reformulation that leverages in-context learning of Large Language Models (LLMs) to revise the edit commands generated by the Doc2Command module. Fig. 15 in the Appendix shows the prompt template comprising of the original user request and the edit command from Doc2Command used with an LLM for this purpose. The output from the LLM is an edit instruction customized for LMM-based document editing. Fig. 3 represents two qualitative examples demonstrating command reformulation and the associated impact on the edited document.

3.3 Generative Document Editing

HTML+CSS as Document Representations: Structured textual representations, such as Hypertext Markup Language (HTML) and Cascading Style Sheets (CSS), present notable advantages in alleviating the challenges associated with generative methods in document editing. Firstly, HTML provides a hierarchical structure that inherently captures the organization and relationships among document elements, facilitating the preservation of structural information. This hierarchical representa-



(a) The reformulated command, by virtue of specificity, is able to achieve the desired edit.

(b) Commands reformulation performs better document grounding due to ambiguities in the generated command and distractors in the document image.

Figure 3: Examples showing commands generated post-Doc2Command and Command Reformulation prompting.

tion enables precise manipulation and control over the layout and arrangement of content, which is essential for maintaining document coherence during the editing process. Secondly, CSS decouples content from presentation, offering a systematic approach to capture stylistic attributes such as fonts, colors, and layouts. This separation of content and style allows for greater flexibility in rendering documents while preserving their underlying structure. Hence, we conceptualize document editing as a text generation task by expressing the document as an HTML+CSS rendering.

Generating HTML+CSS Data: We employ generative large multimodal models (LMMs), specifically GPT-4V and Gemini, to convert both the input as well as ground truth document images into a closely replicated HTML and CSS rendering via constraint-driven prompt engineering. Our experimental setup imposes strict constraints on the generated HTML documents to ensure standardization across class names, adequate utilization of flexbox for layouts, higher preference for embedded CSS, and replacement of visual media with placeholders. Maintaining consistency and coherence across the generated HTML+CSS facilitates fair evaluation.

LMM Prompting: We utilize multimodal prompting of GPT-4V and Gemini by incorporating the set of marks (Yang et al., 2023a) for the grounded RoI bounding boxes extracted by Doc2Command and the edit instruction produced in the Command Reformulation step. Such multimodal prompting guides LMMs to closely adhere to the provided commands while paying special attention to the visual cues specified by the bounding box in the document image. This ensures that the generated edits accurately reflect the intended modifications.

4 Document Editing Evaluation

We perform system output evaluation as follows:

Automated Metrics: Apart from the document metrics reported by Mathur et al. (2023a) for command text generation (Exact Match, ROUGE-L, Word Overlap F1, Action and Component Accuracy %) and RoI bounding box prediction (Top-1 accuracy %), we adapt two novel metrics, specific to HTML document editing:

(1) DOM Tree Edit Distance – Document Object Model (DOM) tree represents the hierarchical structure of the HTML document. Comparing the DOM tree of two HTML documents yields information about their structural differences. We utilize the Zhang-Shasha algorithm (Zhang and Shasha, 1989) to calculate the edit distance between the generated and ground truth DOM trees.

(2) CSS IoU: Cascading Style Sheets (CSS) deal

with the presentation of HTML documents and dictates how they would be rendered. In recreating document images into HTML pages, CSS in the form of property-value pairs of different attributes controls the formatting, style and layout of the rendered HTML document. Sets of property-value pairs from inline CSS and internal CSS selectors are obtained, and the Intersection over Union (IoU) is calculated over these sets to evaluate the similarity between the styles of the edited and ground truth documents. We also evaluate parallel HTML documents using ROUGE-L and Word Overlap F1, applied to the entire document.

Human Evaluation: Every edited document HTML is evaluated by three human evaluators on our three proposed metrics: **(1) Style Replication** assesses whether the styles of the original document are preserved, **(2) Content Replication** evaluates if the textual content of the region of non-interest in the original document HTML is conserved, **(3) Edit Correctness**: judges whether the user's editing intent has been faithfully fulfilled. Each of these metrics yields a binary score, which is averaged across evaluators and then summed to compute a unified score for each document.

5 Experimental Settings

5.1 Data

We utilize the DocEdit-PDF dataset, introduced by Mathur et al. (2023a). The dataset comprises pairs of 17,808 document images, with corresponding user edit requests and ground truth edit commands. Our experiments are conducted on the default data split provided in the official dataset release, wherein the data is partitioned into training, testing, and validation sets in an 8:2:1 ratio. All reported results are based on the test set. The license for the dataset can be found here.

5.2 Implementation Details

Doc2Command Our experiments utilized the Adafactor optimization algorithm with a learning rate of 3×10^{-5} and weight decay set to 1×10^{-5}. The training process spanned 30 epochs with a batch size of 1. The input data was organized into patches of size 16, limiting the maximum number of patches to 1024. The learning rate was scheduled using a cosine scheduler with a warm-up period equivalent to 10% of the iterations within each epoch. For loss computation, we introduced loss weighing factors $\lambda_{\text{text}} = 0.3$ and $\lambda_{\text{seg}} = 1.5$. The sigmoid focal loss was utilized for segmentation with parameters $\alpha = 0.25$ and $\gamma = 2$. Additionally, the decoder included a dropout rate of 0.1.

Command Reformulation and Document Editing: We use `gpt-4` (OpenAI, 2023) and `gemini-pro` (Team et al., 2023) for command reformulation, and `gpt-4-vision-preview/gemini-pro-vision` for document editing. We set the temperature parameter to 0 to ensure deterministic and reproducible experiments and use the default value for all other parameters. The visual grounding and command grounding are obtained by inferring Doc2Command on the test set. The maximum token count for the output is set as 4000.

One limitation of using HTML as a medium to express document edits is that the ground truth post-edit documents only exist as document images, with bounding boxes to indicate edited regions. Therefore, we generate HTML replications of the ground truth post-edit documents using LMMs. To ensure consistency, we use the same prompt details for image-to-HTML conversion as the document editing experiments. Additionally, we prompt the model to pay special attention to the style and content in the bounding box while recreating the document image as an HTML document. We perform human evaluation of the ground truth post-edit HTML documents by comparing them to ground truth images as described in the Metrics subsection and find that style replication score and content replication score are 75.23% and 92.3% (GPT-4V), and 70.14% and 87% (Gemini) respectively, with a Cohen's Kappa score ≥ 0.84 across evaluators and tasks. More implementation details on the metrics (Sec. A.3), computational resources (Sec. A.4, and human evaluations (Sec. A.5) are in the Appendix.

6 Baselines

Command Grounding Baselines: We investigate several command generation baselines to establish performance benchmarks. Initially, we employ Seq2Seq text-only models, including GPT2 (Radford et al., 2019), BART (Lewis et al., 2020), and T5 (Raffel et al., 2020), which exclusively process user text descriptions. Subsequently, we explore the Generator-Extractor paradigm, integrating BERT (Devlin et al., 2019) and DETR (Carion et al., 2020) with autoregressive decoding for command generation. Additionally, we examine Transformer Encoder-Decoder architectures,

System	EM (%)	Word Overlap F1	ROUGE-L	Action (%)	Component (%)
Generator-Extractor	6.6	0.25	0.22	36.7	8.5
GPT2 (Radford et al., 2019)	11.6	0.76	0.76	79.7	27.2
BART (Lewis et al., 2020)	19.7	0.78	0.76	81.2	29.5
T5 (Raffel et al., 2020)	20.4	0.79	0.76	81.4	29.8
BERT2GPT2	7.3	0.37	0.39	45.2	9.2
LayoutLMv3-GPT2	8.7	0.39	0.40	47.6	10.3
CLIPCap (Mokady et al., 2021)	8.5	0.25	0.27	44.5	9.34
DiTCap (Lewis et al., 2006)	23.6	0.81	0.80	82.5	25.5
Multimodal Transformer (Hu et al., 2020)	31.6	0.82	0.83	83.1	32.4
DocEditor (Mathur et al., 2023a)	37.6	0.87	0.83	87.6	40.7
GPT3.5 (Brown et al., 2020)	10.1	0.77	0.77	75.93	73.37
GPT4 (OpenAI, 2023)	14.3	0.78	0.78	81.57	75.03
Doc2Command	39.6	0.87	0.86	85.0	86.1

Table 1: Results for the command generation task. Doc2Command shows the best performance (see Red).

System	Top-1 Acc (%)
ReSC-Large (Yang et al., 2020)	17.04
Trans VG (Deng et al., 2022)	25.34
DocEditor (Mathur et al., 2023a)	36.50
Doc2Command	48.69

Table 2: Results for bounding box detection task. Doc2Command shows the best performance (see Red).

such as LayoutLMv3-GPT2 and BERT2GPT2 (Huang et al., 2022), which combine GPT2 decoders with LayoutLMv3 and BERT encoders, respectively. Furthermore, we investigate Prefix Encoding (Mokady et al., 2021), utilizing learned representations from pre-trained encoders like CLIP (Radford et al., 2021) and DiT (Lewis et al., 2006) as a prefix to the GPT2 decoder network. Additionally, we consider the Multimodal Transformer(Hu et al., 2020), which incorporates multimodal input from user descriptions, visual objects, and document text to generate commands. Moreover, we explore DocEditor (Mathur et al., 2023a), a task-specific baseline employing a Transformer-based multimodal model that decomposes document images into OCR content and object boxes, utilizing multimodal transformers to generate commands. Finally, we compare against GPT3.5 (Brown et al., 2020) and GPT4 (OpenAI, 2023), employing in-context learning by providing three examples of each command type as context to the model for evaluation.**Visual Grounding Baselines:** We consider several baselines for bounding box detection in the context of visual grounding for document editing. Firstly, ReSC-Large (Yang et al., 2020) presents a method for direct coordinates regression in the Region of Interest (RoI) bounding box prediction task. Similarly, TransVG (Deng et al., 2022) offers an alternative approach for direct coordinates regression in RoI bounding box prediction. Additionally, we investigate DocEditor (Mathur et al., 2023a), which employs a comprehensive methodology. DocEditor initially encodes the document image by extracting text through Optical Character Recognition (OCR) and utilizes object detection to capture visual features. Subsequently, transformer-encoded features are fed into a Gated Relational Graph Convolutional Network (R-GCN) to generate a layout graph-aware representation. This representation is then leveraged downstream to perform bounding box regression, facilitating accurate localization of document elements. **Document Editing Baselines:** Certain experimental configurations are employed to investigate the effectiveness of command reformulation and multimodal grounding in harnessing the capabilities of GPT-4V and Gemini as document editing tools. Specifically, visual grounding, command grounding, and command reformulation are selectively excluded from our experiments. In this context, command grounding is supplanted by the unstructured user request, while visual grounding is eliminated by presenting the original document image as the input, thus eliminating the need for explicit visual cues (rendered bounding boxes). Moreover, command reformulation is eliminated by directly utilizing the command generated by the Doc2Command model. Notably, the absence of command grounding renders command reformulation inapplicable (N/A), as the reformulation process relies on refining commands derived from grounded contexts.

7 Results

Edit Request Grounding: Table 1 shows the performance of `DocEditAgent` against contemporary baselines for command generation tasks. `DocEditAgent` achieves an impressive 86.1% accuracy in recognizing document components , outperforming the previous state-of-the-art (SoTA)

Experimental Setting				Automated Evaluation				Human Evaluation			
Method	**VG**	**CG**	**CR**	**ROUGE-L**	**Word Overlap F1**	**Tree Edit Distance**	**CSS IoU**	**SR (%)**	**EC (%)**	**CC (%)**	**Total Score (%)**
GPT-4V Only	✗	✗	N/A	0.406	0.451	24.13	0.245	73.53	27.45	66.77	55.92
GPT-4V +	✓	✗	N/A	0.410	0.460	24.02	0.250	74.28	45.28	68.21	62.59
	✗	✓	✗	0.412	0.458	23.54	0.247	75.02	49.32	68.22	64.19
	✗	✓	✓	0.409	0.455	23.27	0.245	74.87	51.87	69.71	65.49
	✓	✓	✗	0.416	0.461	23.72	0.251	75.14	55.33	69.89	66.79
DocEditAgent	✓	✓	✓	0.417	0.463	23.15	0.252	75.31	57.41	69.14	67.28

Table 3: Results and ablations for end-to-end document editing task using GPT-4V as the base LMM. Here, VG = Visual Grounding, CG = Command Generation, and CR = Command Reformulation. Red represents best performance.

Experimental Setting				Automated Evaluation				Human Evaluation			
Method	**VG**	**CG**	**CR**	**ROUGE-L**	**Word Overlap F1**	**Tree Edit Distance**	**CSS IoU**	**SR (%)**	**EC (%)**	**CC (%)**	**Total Score (%)**
Gemini Only	✗	✗	N/A	0.438	0.542	62.95	0.333	59.64	15.79	61.41	45.61
Gemini +	✓	✗	N/A	0.447	0.551	54.63	0.332	60.12	39.22	65.02	54.79
	✗	✓	✗	0.451	0.544	65.06	0.334	61.92	37.65	64.28	54.62
	✗	✓	✓	0.417	0.510	53.89	0.341	62.52	40.44	67.11	56.69
	✓	✓	✗	0.437	0.554	55.41	0.342	64.12	41.35	66.96	57.48
DocEditAgent	✓	✓	✓	0.454	0.557	52.24	0.367	63.16	44.73	68.42	58.77

Table 4: Results and ablations for end-to-end document editing task using Gemini as the base LMM. Here, VG = Visual Grounding, CG = Command Generation, and CR = Command Reformulation. Red represents best performance.

by 10.7%. We see consistent gains for the exact match accuracy and ROUGE-L score, although comparable performance to SOTA across action accuracy $(\%)$ and word overlap F1. We show significant improvement in component accuracy $(\%)$ over the previous task specific SoTA, 45% points. We attribute this notable improvement to the Doc2Command module, which can effectively comprehend natural language requests and ground them into complex document structures and layouts. Table 2 shows that Doc2Command yields remarkable enhancements in the bounding box detection task with a Top-1 accuracy of 48.69%, surpassing the previous SoTA performance by 12.19%, which further signifies our system's effectiveness in accurately grounding edit requests to document images.

Generative Document Editing: Table 3 and 4 shows the results for end-to-end document editing task with GPT-4V and Gemini as the base LMMs respectively. We observe that Doc2Command and Command Reformulation prompting are critical components as removing either severely deteriorates performance across automated and human evaluations. We observe ~2-3 % decline in Edit Correction when command reformulation prompting is removed (in both settings: with or without visual grounding) . Visual grounding assists by localising the edit region, which can be demonstrated by an improvement of $\sim 18 - 23\%$ when GPT-4V is prompted with visual grounding.

Significant performance gains across Tree Edit Distance and CSS IoU indicate the ability of GPT-4V and Gemini to consistently recreate non-RoI parts of the document, proving the effectiveness of editing HTML and CSS directly. The experiment setting with no multimodal grounding performs worst, while multimodal grounding with command reformulation improves editing correctness (EC) by 29.96%(GPT-4V)/28.94%(Gemini) and overall human evaluation score by 11.36%(GPT-4V)/13.16%(Gemini).

Fig 5-13 show qualitative examples of document editing by `DocEditAgent` for diverse edit requests such as spatial alignment, component placement, text paraphrasing and applying special effects which involve manipulating and rendering different document elements such as text, tables, figures and lists.

8 Conclusion

We introduce the `DocEditAgent` framework for end-to-end document editing. DocEditAgent draws on Doc2Command, a multi-task multimodal model that visually localizes user requests in the document image and generates edit commands, which are further refined using Command Reformulation prompting. `DocEditAgent` uses LMMs multimodal prompting with request grounding and edit instructions to perform generative editing of the HTML+CSS structure of documents, showcasing remarkable performance improvements across editing accuracy, command generation, and RoI detection. Future work will aim to enhance the framework's adaptability to diverse document types, including multi-page documents.

9 Ethics Statement

We utilize the publicly available DocEdit-PDF corpus for this research without introducing new annotations. We use publicly available API-accessible LMMs and LLMs for our experiments. The identity of the human evaluators is confidential and private. We do not utilize any PII at any step in our experiments. The intended applications of our work are strictly limited to the document editing domain. We refer users to relevant works by (Kumar et al., 2024; Cui et al., 2024; Luu et al., 2024) to understand risks and some mitigation strategies for LLM safety.

10 Limitations

1. **Document Recreation** The DocEdit Corpus (Mathur et al., 2023a) has documents only as document images. Pixel level manipulation of text-dense image is a challenge, hence we prompt LMMs to produce faithful HTML+CSS recreations. The HTML+CSS documents are close but not identical to the original document images.

2. **Visual Elements** `DocEditAgent` is constrained with generating edited documents as HTML+CSS documents. Complex visual elements such as charts and figures cannot be generated using simple HTML and CSS. Moreover, the transformer backbone used in Doc2Command is pre-trained primrarily on text-dominant document images and has limitations in grounding requests manipulating these visual elements.

3. **Large Multimodal Models** Our work utilizes API-accessible Large Multimodal Models (LMMs). Model APIs have an associated cost which depends on the token count in the request and model response, image resolution and dimensions. These API based models are also prone to performance fluctuations.

References

Tim Brooks, Aleksander Holynski, and Alexei A Efros. 2023. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18392–18402.

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.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-end object detection with transformers. *CoRR*, abs/2005.12872.

Yu Cheng, Zhe Gan, Yitong Li, Jingjing Liu, and Jianfeng Gao. 2020. Sequential attention gan for interactive image editing. In *Proceedings of the 28th ACM international conference on multimedia*, pages 4383–4391.

Tianyu Cui, Yanling Wang, Chuanpu Fu, Yong Xiao, Sijia Li, Xinhao Deng, Yunpeng Liu, Qinglin Zhang, Ziyi Qiu, Peiyang Li, Zhixing Tan, Junwu Xiong, Xinyu Kong, Zujie Wen, Ke Xu, and Qi Li. 2024. Risk taxonomy, mitigation, and assessment benchmarks of large language model systems.

Jiajun Deng, Zhengyuan Yang, Tianlang Chen, Wengang Zhou, and Houqiang Li. 2022. Transvg: End-to-end visual grounding with transformers.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*.

Weixi Feng, Wanrong Zhu, Tsu-Jui Fu, Varun Jampani, Arjun Reddy Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. 2023. Layoutgpt: Compositional visual planning and generation with large language models. *ArXiv*, abs/2305.15393.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2022. Pal: Program-aided language models. *ArXiv*, abs/2211.10435.

Liu He, Yijuan Lu, John Corring, Dinei A. F. Florêncio, and Cha Zhang. 2023. Diffusion-based document layout generation. In *IEEE International Conference on Document Analysis and Recognition*.

Ronghang Hu, Amanpreet Singh, Trevor Darrell, and Marcus Rohrbach. 2020. Iterative answer prediction with pointer-augmented multimodal transformers for textvqa. In *Proceedings of the IEEE/CVF conference*

on computer vision and pattern recognition, pages 9992–10002.

Siyuan Huang, Zhengkai Jiang, Hao Dong, Yu Jiao Qiao, Peng Gao, and Hongsheng Li. 2023. Instruct2act: Mapping multi-modality instructions to robotic actions with large language model. *ArXiv*, abs/2305.11176.

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*.

Wentao Jiang, Ning Xu, Jiayun Wang, Chen Gao, Jing Shi, Zhe Lin, and Si Liu. 2021a. Language-guided global image editing via cross-modal cyclic mechanism. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2115–2124.

Yuming Jiang, Ziqi Huang, Xingang Pan, Chen Change Loy, and Ziwei Liu. 2021b. Talk-to-edit: Fine-grained facial editing via dialog. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13799–13808.

K. J. Joseph, Prateksha Udhayanan, Tripti Shukla, Aishwarya Agarwal, Srikrishna Karanam, Koustava Goswami, and Balaji Vasan Srinivasan. 2024. Iterative multi-granular image editing using diffusion models. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 8107–8116.

Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. 2023. Imagic: Text-based real image editing with diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6007–6017.

Katya Kudashkina, Patrick M. Pilarski, and Richard S. Sutton. 2020. Document-editing assistants and model-based reinforcement learning as a path to conversational ai. *ArXiv*, abs/2008.12095.

Ashutosh Kumar, Sagarika Singh, Shiv Vignesh Murty, and Swathy Ragupathy. 2024. The ethics of interaction: Mitigating security threats in llms.

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

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

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Bowen Li, Xiaojuan Qi, Thomas Lukasiewicz, and Philip HS Torr. 2020. Manigan: Text-guided image manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7880–7889.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

Tzu-Hsiang Lin, Alexander Rudnicky, Trung Bui, Doo Soon Kim, and Jean Oh. 2020. Adjusting image attributes of localized regions with low-level dialogue. *arXiv preprint arXiv:2002.04678*.

Huan Ling, Karsten Kreis, Daiqing Li, Seung Wook Kim, Antonio Torralba, and Sanja Fidler. 2021. Editgan: High-precision semantic image editing. *Advances in Neural Information Processing Systems*, 34:16331–16345.

Quan Khanh Luu, Xiyu Deng, Anh Van Ho, and Yorie Nakahira. 2024. Context-aware llm-based safe control against latent risks.

Puneet Mathur, Rajiv Jain, Jiuxiang Gu, Franck Dernoncourt, Dinesh Manocha, and Vlad I Morariu. 2023a. Docedit: language-guided document editing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 1914–1922.

Puneet Mathur, Rajiv Jain, Ashutosh Mehra, Jiuxiang Gu, Franck Dernoncourt, Quan Tran, Verena Kaynig-Fittkau, Ani Nenkova, Dinesh Manocha, Vlad I Morariu, et al. 2023b. Layerdoc: layer-wise extraction of spatial hierarchical structure in visually-rich documents. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3610–3620.

Ron Mokady, Amir Hertz, and Amit H. Bermano. 2021. Clipcap: CLIP prefix for image captioning. *CoRR*, abs/2111.09734.

Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. 2021. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

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.

Jing Shi, Ning Xu, Trung Bui, Franck Dernoncourt, Zheng Wen, and Chenliang Xu. 2020. A benchmark and baseline for language-driven image editing. In *Proceedings of the Asian Conference on Computer Vision*.

Carole H Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M Jorge Cardoso. 2017. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support: Third International Workshop, DLMIA 2017, and 7th International Workshop, ML-CDS 2017, Held in Conjunction with MICCAI 2017, Québec City, QC, Canada, September 14, Proceedings 3*, pages 240–248. Springer.

D'idac Sur'is, Sachit Menon, and Carl Vondrick. 2023. Vipergpt: Visual inference via python execution for reasoning. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11854–11864.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Narek Tumanyan, Michal Geyer, Shai Bagon, and Tali Dekel. 2023. Plug-and-play diffusion features for text-driven image-to-image translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1921–1930.

Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. 2023a. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v.

Zhengyuan Yang, Tianlang Chen, Liwei Wang, and Jiebo Luo. 2020. Improving one-stage visual grounding by recursive sub-query construction.

Zongyuan Yang, Baolin Liu, Yongping Xiong, Lan Yi, Guibin Wu, Xiaojun Tang, Ziqi Liu, Junjie Zhou, and Xing Zhang. 2023b. Docdiff: Document enhancement via residual diffusion models. *Proceedings of the 31st ACM International Conference on Multimedia*.

Kaizhong Zhang and Dennis Shasha. 1989. Simple fast algorithms for the editing distance between trees and related problems. *SIAM J. Comput.*, 18:1245–1262.

A Appendix

A.1 Examples

Fig. 4 represents 6 examples of our model's performance on the test set. Subfigures (a), (b), and (c) represent correctly inferred examples, and (d), (e), and (f) represent incorrectly inferred examples. With each example, the figure explains the capability or limitation of our system demonstrated by the example.

The examples presented in `Table 5` showcase six instances of commands generated from user requests. However, the first three examples highlight situations where our model deviates from replicating the ground truth command. A detailed analysis of these errors is provided below:

1. In the first example, while the generated command achieves the desired document edit, the ground truth command exhibits more efficiency as it achieves the same outcome with fewer changes.

2. The second example illustrates an incorrect command generated by the model, wherein it mistakes a "split" action for a "replace" action. Consequently, the edited document does not align with the intended user request.

3. In the third example, the model considers the logo as a visual element, contrary to the ground truth, which recognizes it as a textual element within the document.

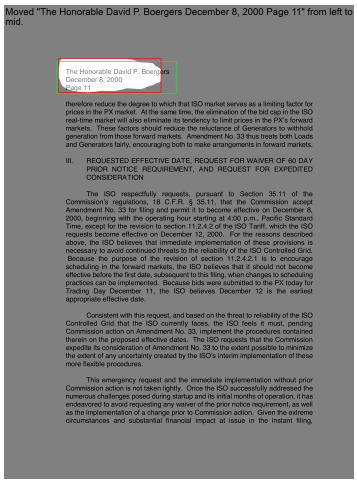
Examples of end to end document editing are shown in Fig 5-13. Each of these figures illustrates the user request and document image, followed by multimodal grounding using Doc2Command, command reformulation and finally the rendered HTML+CSS document.

A.2 Prompt Templates

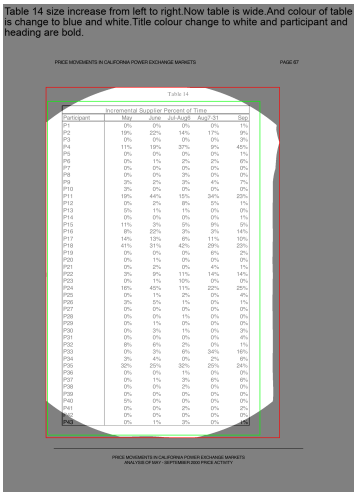
Fig 14, 15 and 16 represent the prompt templates used in different steps of our pipeline, with Large Language Models or Large Multimodal Models.

User Request		ACTION_PARA	COMPONENT_PARA	INITIAL_STATE	FINAL_STATE
Change the date "December 1, 2000" to December 11, 2020	*Predicted*	replace	text	December 1, 2000	December, 11, 2000
	Ground Truth	modify	text	1, 2000	11, 2000
2-3 lines of text in the paragraph "(p) Issues, obtain" are changed to four separate bullet points. Bullet a. "any department or agency of the United States", b."from other agencies of the state", c. "from any private company" and d. "any insurance or guarantee to"	*Predicted*	replace	bullet	dotted	4 bullet points
	Ground Truth	split	text	paragraph	split
Moved logo from left to right.	*Predicted*	move	image	left	right
	Ground Truth	move	text	left	right
Delete all data from table "Tabela 15 Układ pasywów bilansu jednostek, z wyłączeniem banków—"	*Predicted*	delete	text	in table	removed
	Ground Truth	delete	text	in table	removed
Added page number 4 at the footer of the page.	*Predicted*	add	text footer	none	Page 4
	Ground Truth	add	text footer	none	Page 4
removed the space after the heading fundamental corrective measures.	*Predicted*	merge	text	not merged	merged; heading with text
	Ground Truth	merge	text	not merged	merged; heading with text

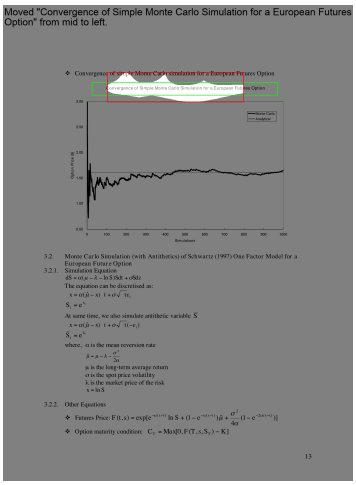
Table 5: Examples of command generation in Doc2Command. Correct command parameters are highlighted in green, and incorrect command parameters are highlighted in red.



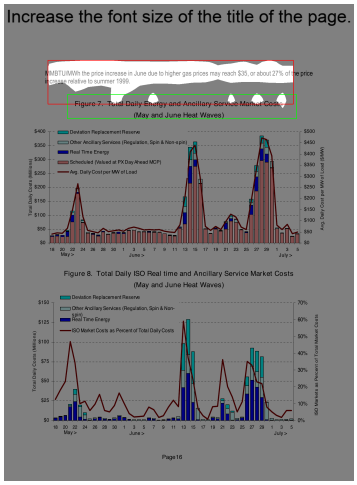
(a) Bounding Box with high IOU: capability to read and recognise text from request in the document.



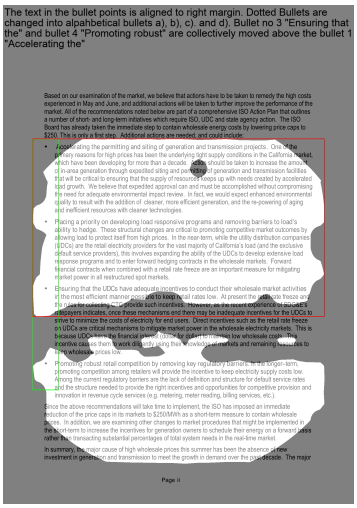
(b) Bounding Box with high IOU: capability to recognise elements such as tables.



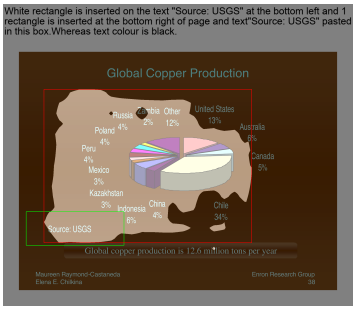
(c) Bounding Box with high IOU: When given two elements with the same text, capability to localize based on position reference.



(d) Bounding Box with low IOU: Ambiguity in the page's title.



(e) Bounding Box with low IOU: Mask highlights the points that have been bulleted but not the bullets exclusively.



(f) Bounding Box with low IOU: edit request involves text in visual elements

Figure 4: Examples of segmentation outputs and bounding boxes. The bright white areas represent segmentation outputs. Green boxes represent ground truth bounding boxes, and red boxes represent the inferred bounding boxes.

A.3 Additional Evaluation Metrics

We adapt these metrics from (Mathur et al., 2023a).

Command Grounding Metrics

- Exact Match: Percentage of generated commands that exactly match the ground truth commands.
- Word Overlap F1: Measures the F1 of the word overlap score between the generated and ground truth commands.

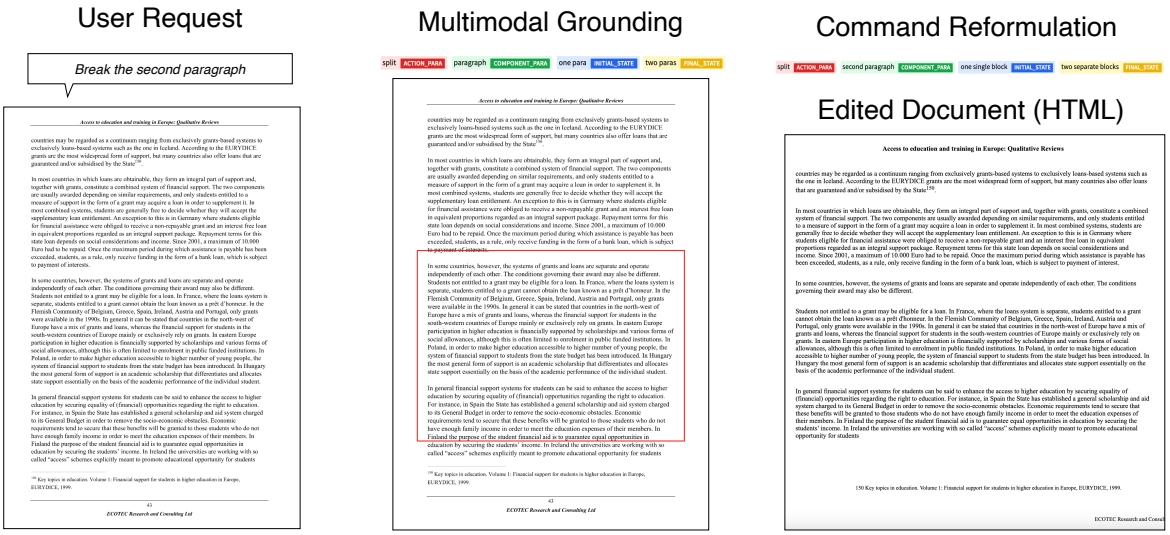


Figure 5: Example of document editing request, corresponding multimodal grounding, command reformulation and edit generation.

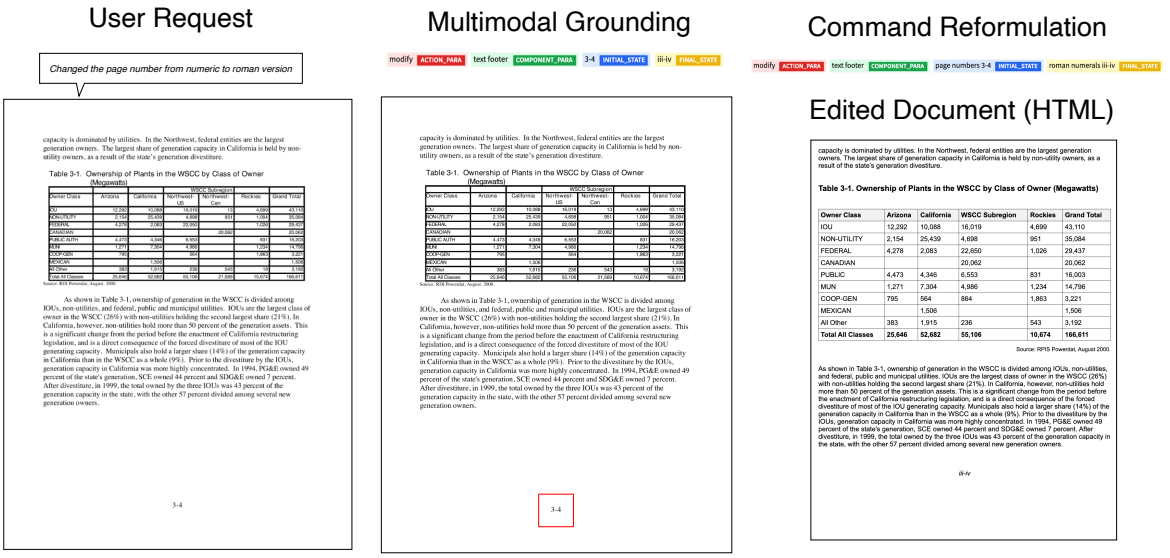


Figure 6: Example of document editing request, corresponding multimodal grounding, command reformulation and edit generation.

- ROUGE-L: Evaluates the longest common subsequence of words between the generated and ground truth commands.
- Action (%): Percentage of commands with exact matches in the action parameter.
- Component (%): Percentage of commands with exact matches in the component parameter.

Visual Grounding Metrics

- Top-1 Accuracy: Measures the accuracy of visual grounding, where a match is considered when the Jaccard overlap is greater than or equal to 0.5.

A.4 Computational Resources

Table 6 gives an overview of computational resources used in our experiments for Doc2Command.

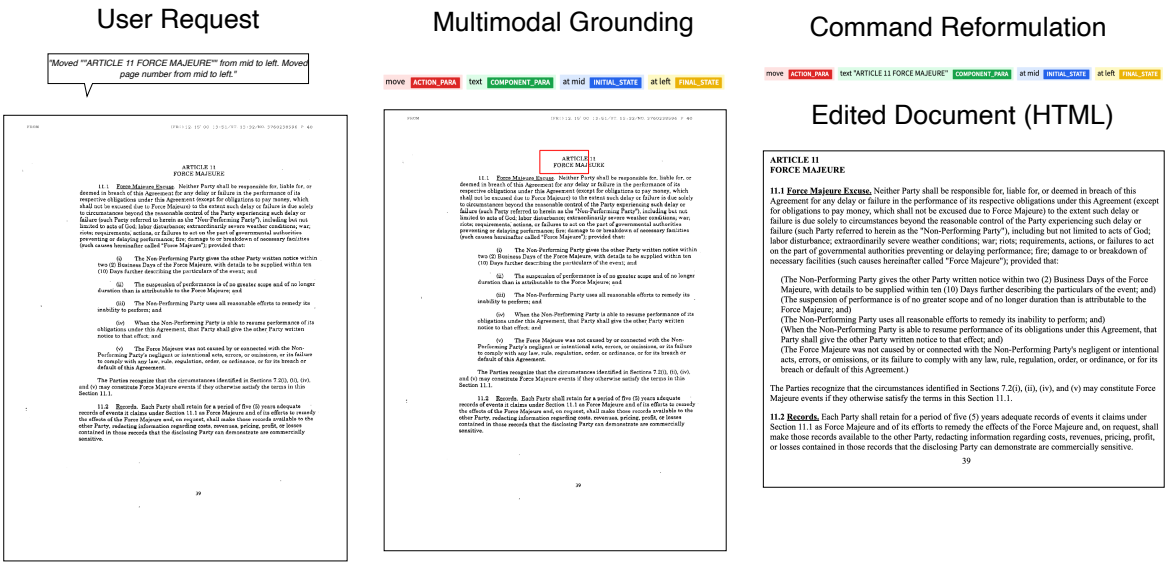


Figure 7: Example of document editing request, corresponding multimodal grounding, command reformulation and edit generation.

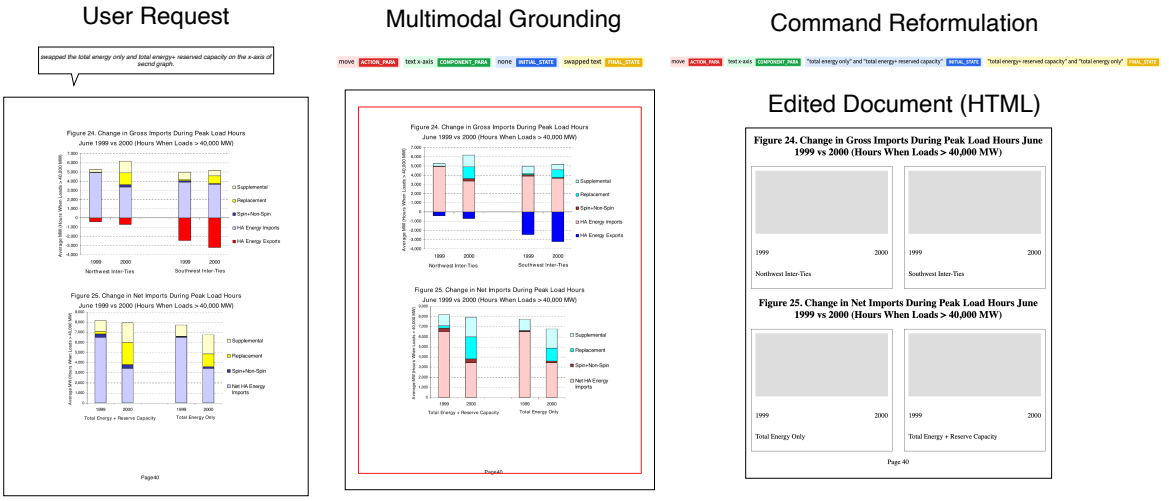


Figure 8: Example of document editing request, corresponding multimodal grounding, command reformulation and edit generation.

Parameter	Value
GPU Hours	100
Number of Parameters	3M
GPU Specification	NVIDIA GeForce RTX 2080 Ti
Number of GPUs	1

Table 6: Overview of computational resources required in training and experimenting with Doc2Command.

A.5 Human Evaluation Instructions

The human evaluators are college graduates expected to have basic knowledge of working with PDF documents. They are provided with a comprehensive rubric for evaluation and a set of examples to guide to demonstrate the evaluation process. Fig 17 shows the UI used by human evaluators, and table 7 shows a concise version of the evaluation rubric annotators are expected to refer for each sample. Each annotator examines the renderings of the edited HTML document generated

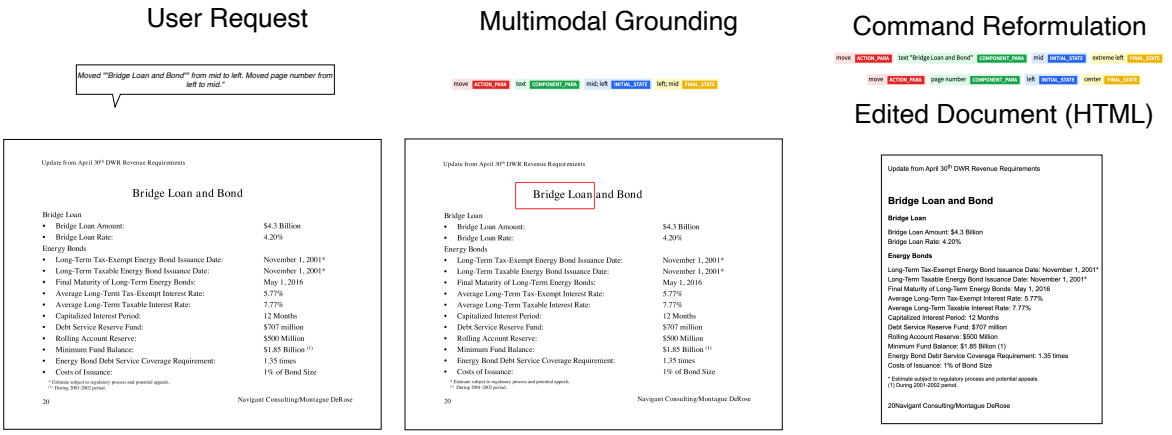


Figure 9: Example of document editing request, corresponding multimodal grounding, command reformulation and edit generation.

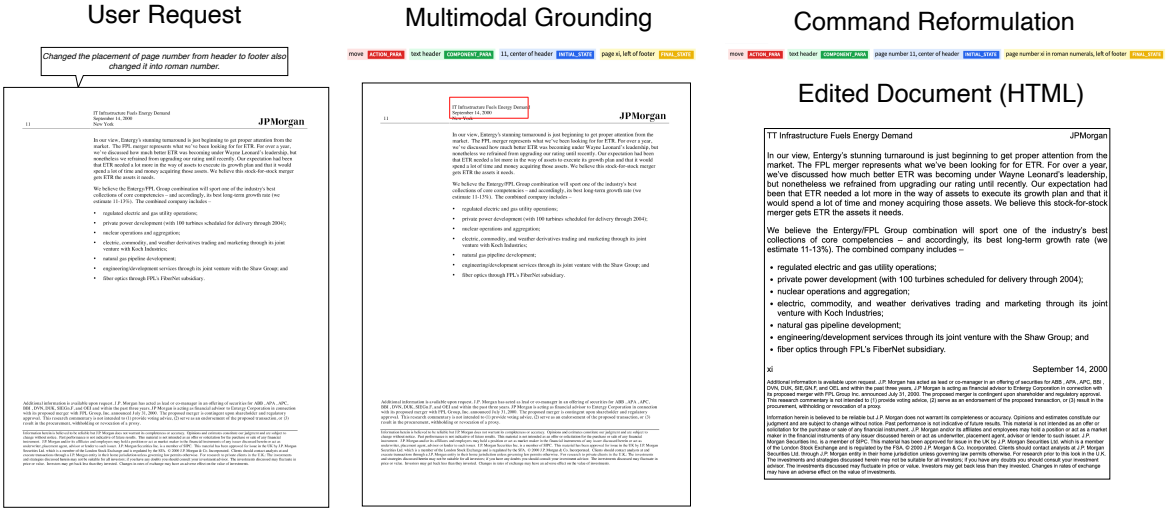


Figure 10: Example of document editing request, corresponding multimodal grounding, command reformulation and edit generation.

by `DocEditAgent` and the ground truth pre- and post-edit document images. Evaluator are compensated well above average wages according to their geographical locations for their contributions.

A.6 Methodology: Doc2Command Command Generation

The input image is represented as $I \in \mathbb{R}^{H \times W \times C}$, where H and W are the re-scaled height and width of the image, and C is the number of channels. To prepare the image as input into the transformer style encoder, the image is divided into patches, denoted by $P_{i,j} \in \mathbb{R}^{p \times P \times C}$, where p is the patch size and i, j index the patches. Each patch is flattened to obtain a vector of pixel values: $V_{i,j} \in \mathbb{R}^{P^2 \times C}$. The flattened patches are then fed into the image encoder ($\mathcal{E}_I$) to generate patch encodings $Z_I = \{Z_{i,j} \forall i, j\}, Z_I \mathbb{R}^{N \times d1}$ such that $Z_{i,j} = \mathcal{E}_I(V_{i,j})$, where N is the number of patches and $d1$ is the encoder dimension. The patch embeddings generated by the encoder serve as input to the text decoder, which auto-regressively generates a sequence of r tokens, CT representing the command text as $CT = \mathcal{D}_T(Z)$, where $CT = \{s_1, s_2 \ldots s_r\}$. The taxonomy of actions includes Add, Delete, Copy, Move, Replace, Split, Merge, and Modify.

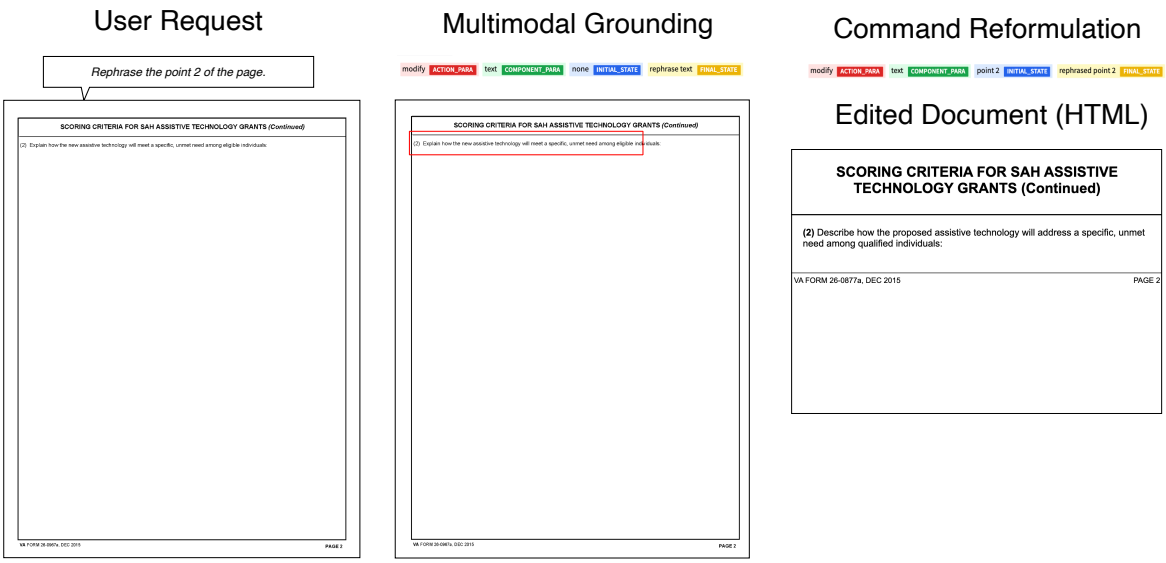


Figure 11: Example of document editing request, corresponding multimodal grounding, command reformulation and edit generation.

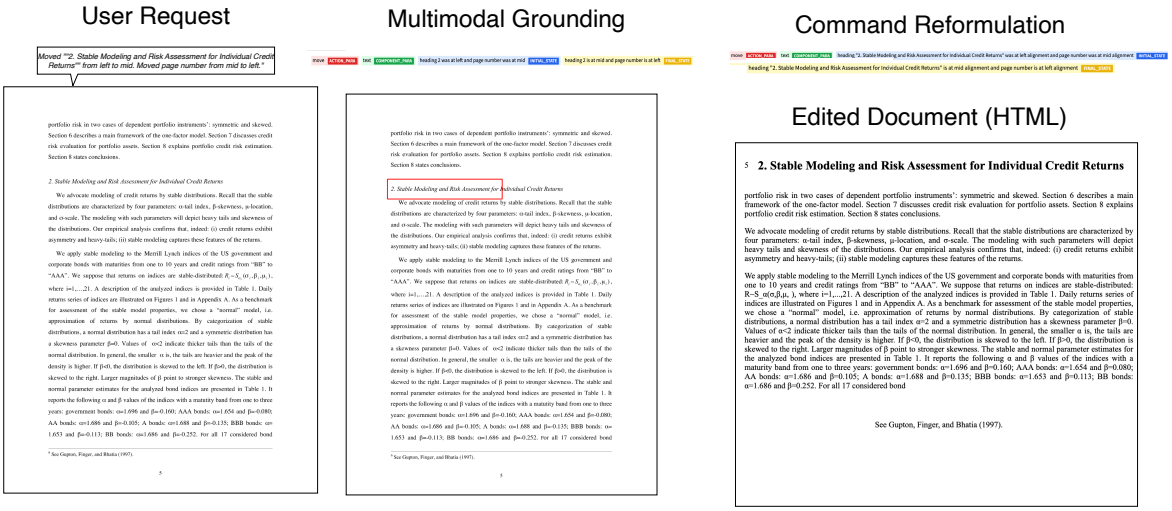


Figure 12: Example of document editing request, corresponding multimodal grounding, command reformulation and edit generation.

A.7 Methodology: Doc2Command Multimodal Grounding

. A point-wise linear layer is applied to the patch encoding $Z \in \mathbb{R}^{N \times D}$ to produce patch-level class logits $Z_{\text{lin}} \in \mathbb{R}^{N \times K}$. The sequence is then reshaped into a 2D feature map $S_{\text{lin}} \in \mathbb{R}^{H/P \times W/P \times K}$ and bilinearly upsampled to the original image size $S \in \mathbb{R}^{H \times W \times K}$. A softmax is applied to the class dimension to obtain the final segmentation map. A set of learnable class embeddings $C \in \mathbb{R}^{K \times d_2}$ is introduced, where K is the number of classes ($K = 3$ for our model), and d_2 is the mask-transformer dimension. Each class embedding is initialized randomly and assigned to a single semantic class. It is used to generate the class mask. The mask-transformer processes the class embeddings jointly with patch encodings $Z_I \in \mathbb{R}^{N \times D}$ such that $C, Z_M = \mathcal{D}_I(C_0, Z_I)$. The mask transformer generates K masks by computing the scalar product between L2-normalized patch embeddings $Z_M \in \mathbb{R}^{N \times d2}$ and class em-

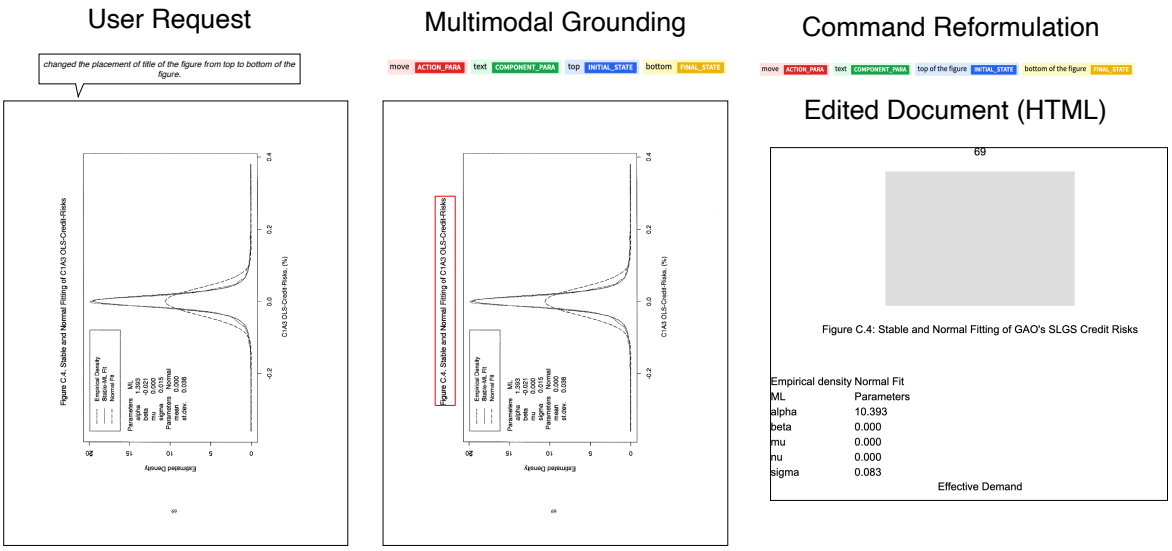


Figure 13: Example of document editing request, corresponding multimodal grounding, command reformulation and edit generation.

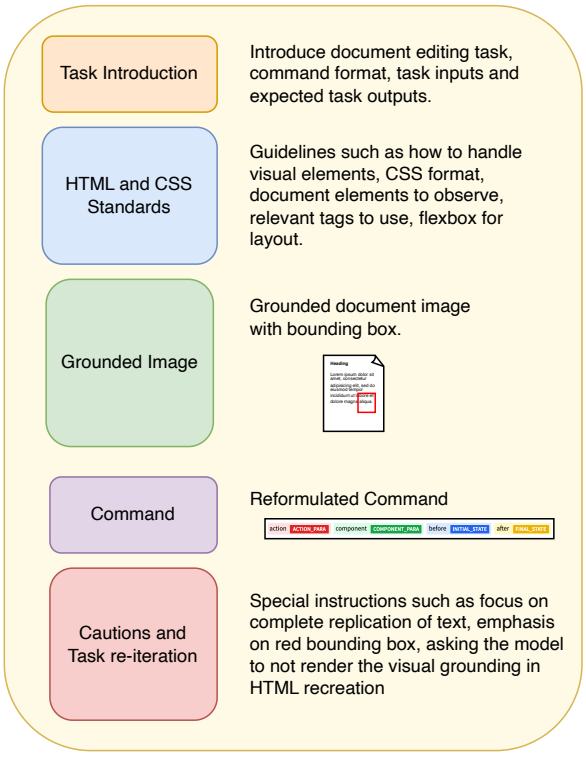


Figure 14: Template of prompt used for document editing using a suitable LMM and multimodally grounded edit request.

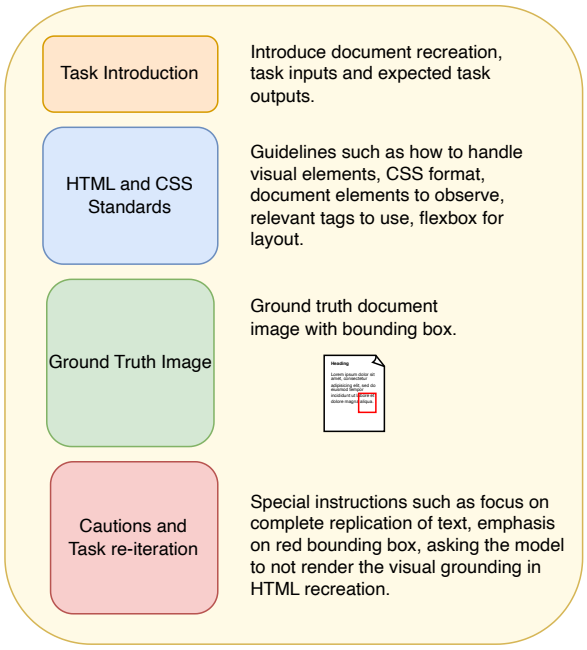


Figure 15: Template of prompt used for generating ground truth document edits from post-edit, visually grounded document images.

beddings $C \in \mathbb{R}^{K \times d2}$ output by the decoder as $\mathcal{M}_I = Z_M \cdot C^T$. The set of class masks is reshaped into a 2D mask $S_I \in \mathbb{R}^{H/P \times W/P \times K}$ and bilinearly upsampled to the image size to obtain a feature map $S \in \mathbb{R}^{H \times W \times K}$. A softmax is then applied to the class dimension, followed by layer normalization to obtain pixel-wise class scores, forming the final segmentation map. The mask sequences are softly exclusive to each other, i.e., $\sum_{k=1}^{K} S_{i,j,k} = 1$ for all $(i, j) \in H \times W$. The Region of Interest (RoI) is represented by the bounding box $[x, y, h, w]$.

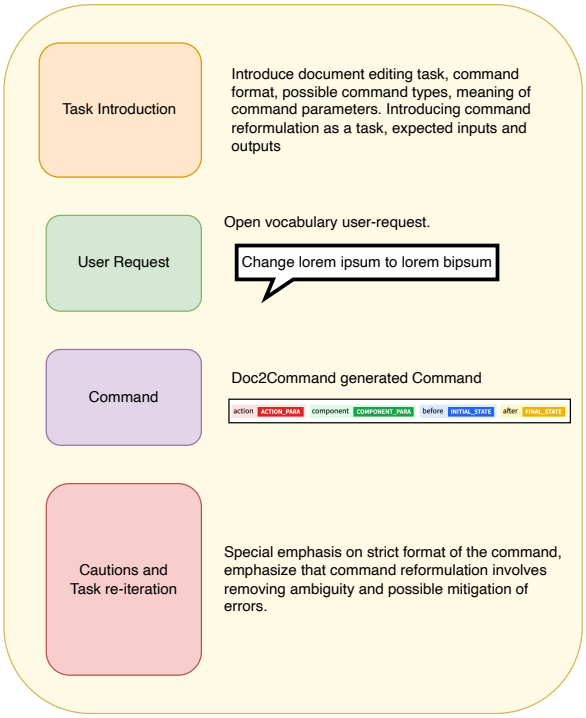


Figure 16: Template of prompt used for reformulating the Doc2Command generated command using an LLM.

Figure 17: UI used by annotators for human evaluation.

Option	Criteria
Content Replication	You should check the Content Completeness (score=1) option if **all** of the following apply: ✓ Elements to be modified are included in the recreation. ✓ At least 80% of textual content has been included in the recreation. ✓ Visual content like figures or charts, if present in the original document are supplanted by placeholders. Further, you should not check the Content Completeness (score=0) option if **any** of the following apply: ✗ Elements to be modified are not included in the recreation. ✗ If the model replaces original text with fillers like *Lorem Ipsum* or hallucinates the document text by a margin of > 20%. Refer to the example set in case of any confusion to understand different case scenarios for Content Completeness.
Style Replication	You should check the Style Replication (score=1) option if **most** of the following apply: ✓ Layout of the elements is correct. ✓ Number of columns the page is divided into. ✓ Position of the text blocks is correct. ✓ Presence of headers/footers. ✓ Alignment and relative placement of elements like dates, page numbers, headings, etc. ✓ Relative text size of different elements is correct. (Example: headings are larger than the text). ✓ Special text like bold/italics/highlight/underline is consistent with the original document. ✓ Relevant elements such as tables, lists or form elements have been used in HTML for document recreation. Each sample contains numerous elements, so you must verify if these rules apply to every individual element before making a decision on if a significant majority of elements are correctly styled. Please refer to the provided example set to understand the acceptable level of deviation for a document to receive a score of 1 for Style Replication.
Edit Correctness	Carefully review the edit request and examine the pre-change document image. As an annotator, your task is to evaluate what the desired change should look like based on the provided instructions. Pay close attention to specific details and elements mentioned in the request. Consider the overall context and purpose of the document to ensure that your interpretation aligns with the user's intention. By thoroughly understanding the pre-change state and the requested modifications, you will be able to accurately assess the changes and ensure they are implemented correctly. This detailed evaluation is crucial for maintaining the quality and consistency of the document. You should check the Edit Correctness (score=1) option if the following apply: ✓ Changes made in the region of interest marked in the ground truth post-edit document image have been EXACTLY replicated in the HTML+CSS rendering. ✓ Changes made in the HTML+CSS rendering are consistent with the original user request. Dealing with conflicts: ✓ Ambiguous user intention: change is consistent with the user request (i.e. naively fulfills the expectation) but not exactly the same as the ground truth post-edit image. – Examples of such conflicts include: element to be modified is ambiguous, or desired change can be reasonably interpreted in multiple ways, score it as 1. ✗ Incomplete modification: If the modified HTML+CSS document implements a modification that does not complete the scope of the original document request or doesn't reasonably replicate the changes demonstrated in the ground truth post-edit document image, score it as 0.
Star	Use the star option if a sample is extremely hard to annotate under any of the above-mentioned categories (low confidence examples) OR if the example demonstrates a unique capability of our document editing system.

Table 7: Concise Evaluation Criteria for Human Evaluation