DOC2COMMAND: FURTHERING LANGUAGE GUIDED DOCUMENT EDITING

Manan Suri[‡], Puneet Mathur[†], Ramit Sawhney[*], Preslav Nakov[*], Dinesh Manocha[‡]
[†]Adobe Research, [*] MBZUAI Abu Dhabi, [‡]University of Maryland College Park
manansuri27@gmail.com, puneetm@adobe.com

ABSTRACT

Language guided document editing is a novel task that includes generating a machine parsable command and a bounding box from an open vocabulary user request. This paper introduces Doc2Command, a multi-task, multimodal model that unifies the document and user request into a singular visual modality and utilises a transformer base image encoder-text decoder to generate the command text. Additionally, it reconceptualises bounding box detection as a segmentation task and employs a mask transformer operating on the image encoder. Doc2Command surpasses baseline models in command text generation, demonstrating significant performance improvements ranging from 2-33% for exact matched commands. It also improves on the bounding box detection task on existing baselines by a margin of 12.19-31.65%.

1 INTRODUCTION

In today's dynamic digital landscape, the pervasive use of digital documents for diverse purposes, ranging from business productivity tasks to customer communication strategies, underscores the indispensability of efficient document editing tools. Mathur et al. introduced the DocEdit dataset and a novel task aimed at furthering language guided document editing. The task focuses on generating an executable command from a linguistic user request to edit a document in accordance with the user's intent. Given a document D and a user request $W = [t_1, t_2, \ldots, t_l]$ representing a sequence of n tokens, our objective is to predict the executable command C. The command format is specified as: `ACTION(<Component>, <Attribute>, <Initial State>, <Final State>, [x, y, h, w])`. The taxonomy of actions includes Add, Delete, Copy, Move, Replace, Split, Merge, and Modify. Arguments follow, detailing document components, attributes, initial and final states, and the Region of Interest (RoI) represented by the bounding box $[x, y, h, w]$. Here, (x, y) refers to the top-left coordinate, while h and w denote the height and width of the bounding box. The task encompasses end-to-end command generation along with bounding box prediction grounded in the document image.

2 METHODOLOGY

At the outset, Doc2Command strategically position the user request by rendering it on the top of the document image. This approach allows for a more flexible integration of language and vision inputs, allowing both the user request and the document image to be processed jointly via the visual modality. We use a pre-trained Vision Transformer (ViT) Dosovitskiy et al. (2021) style encoder from Lee et al., that has been pre-trained with a text decoder on a screenshot parsing and masked document image modelling objective. . Instead of scaling the input image to a pre-defined resolution, we adjust the scaling factor such that the maximum number of fixed-size patches that fit in the sequence length are extracted to mitigate problems caused by extreme aspect ratios.

The patch embeddings generated by the encoder serve as an input to both the text decoder and the mask transformer in a multi-task setup. The text decoder is finetuned to generate the command text in the specified format. Simultaneously, the patch embeddings are also fed into a mask transformer. The mask transformer is a DETR style decoder Carion et al. (2020) that is fine-tuned using a combination of focal loss and dice loss. We aim to perform segmentation of the document image into three distinct classes: 1. The region of Interest, 2. The user request (which had previously been rendered into the document image), and 3. The remaining document. Learnable class tokens for each of these classes are fed into the mask transformer. The L2 normalised class embeddings and patch embeddings from the mask transformer are used to generate masks by computing a scalar dot product. The segmentation mask is used to derive the bounding box during inference.

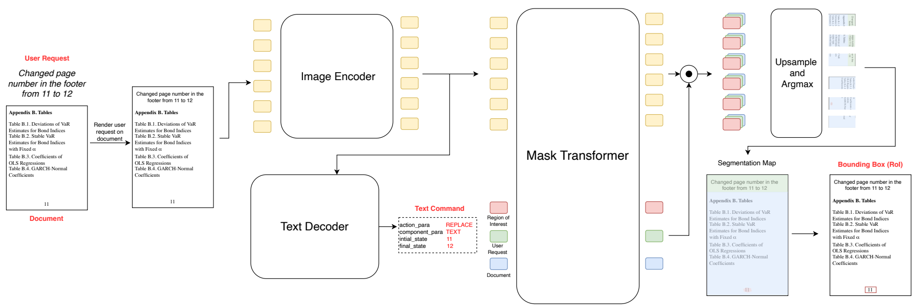


Figure 1: Overview of our proposed system: Doc2Command.

3 Results

We compare our results with various baselines for both command generation (Table 1) and bounding box detection (Table 2). For the command generation task, we greatly outperform existing baselines on recognising the `component`, 86.1% vs 40.7% (previous SoTA). This is a good indication of our model's capabilities of understanding document structures and layouts. In contrast, we perform similar to existing SoTA on recognising the `action`: however, it is interesting to note that text-only baselines such as T5 also perform fairly well on this task since recognising the action is a simpler task. This is helpful in putting into context our model's ability to parse document structure and infer text from the visual modality without additional tools such as OCR. We show great improvements over existing SoTA on the bounding box detection task. We stand at a Top-1 Acc (Jaccard overlap ≥ 0.5) of 48.69% compared to previous SoTA, 36.50%.

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. (2023)	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 and comparison for the command generation task

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

Table 2: Results and comparison for bounding box recognition.

URM Statement

The authors acknowledge that at least one key author of this work meets the URM criteria of ICLR 2024 Tiny Papers Track.

References

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. Language models are few-shot learners, 2020.

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

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

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

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. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

Ronghang Hu, Amanpreet Singh, Trevor Darrell, and Marcus Rohrbach. Iterative answer prediction with pointer-augmented multimodal transformers for textvqa. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9992–10002, 2020.

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

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

David D. Lewis, Gady Agam, Shlomo Engelson Argamon, Ophir Frieder, David A. Grossman, and Jefferson Heard. 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*, 2006.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7871–7880, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL `https://aclanthology.org/2020.acl-main.703`.

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

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

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

OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023. URL https://api.semanticscholar.org/CorpusID:257532815.

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

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. Learning transferable visual models from natural language supervision, 2021.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Carole H Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M Jorge Cardoso. 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*, pp. 240–248. Springer, 2017.

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

A Appendix

A.1 Model Architecture

We use Pix2Struct as our base image encoder and text decoder. Specifically, we use the `google/pix2struct-textcaps-base` implementation from HuggingFace.

Mask-Transformer We approach the detection of bounding boxes through the lens of a segmentation task. Given the bounding boxes for the region of interest, and the rendered user request, we create create ground truth segmentation maps with three classes: 1. the Region of Interest, 2. rendered user request , and 3. the remaining document. We utilise a DETR Carion et al. (2020) style transformer as a mask transformer.

We introduce a set of learnable class embeddings $C \in \mathbb{R}^{K \times e}$, where K represents the number of classes (set as $K = 3$ in our model) and e denotes the mask-transformer dimension. Each class embedding undergoes random initialization and is associated with a single semantic class, serving the purpose of generating the class mask. These class embeddings are processed concurrently with patch encodings $Y_i \in \mathbb{R}^{N \times D}$ through the mask-transformer.

$$C, Y_m = \mathcal{D}_I(C_0, Y_i) \tag{1}$$

The mask transformer produces K masks by computing the scalar product between L2-normalized patch embeddings $Y_m \in \mathbb{R}^{N \times e}$ and class embeddings $C \in \mathbb{R}^{K \times e}$ output by the decoder:

$$\mathcal{B}_i = Y_m \cdot C^T \tag{2}$$

The collection of class masks is reshaped into a 2D mask $Z_i \in \mathbb{R}^{H/P \times W/P \times K}$ and bilinearly upsampled to match the image size, yielding a feature map $S \in \mathbb{R}^{H \times W \times K}$. Subsequently, a softmax operation is applied along the class dimension, followed by layer normalization, to derive pixel-wise class scores, thereby forming the final segmentation map. The mask sequences exhibit soft exclusivity, i.e., $\sum_{k=1}^{K} Z_{i,j,k} = 1$ for all $(i, j) \in H \times W$.

The mask transformer possesses an embedding dimension of 768, comprising 12 layers and 12 attention heads within each layer. The linear layer's dimension is set at 256.

A.2 Data

We use the DocEdit-PDF dataset released by Mathur et al.. The dataset offers a collection of document image-user edit request pairs, accompanied by corresponding ground truth edit commands. Each edit request is associated with an executable command that can be replicated within a real-world document editing software environment. The dataset encompasses approximately 17,808 scanned PDF documents, featuring edits conducted on publicly accessible PDF documents. It includes a varied array of edit operations (add, delete, modify, split, merge, replace, move, and copy) as well as diverse reference types (direct references, object references, and text references) as provided by users. We encourage readers to refer to Mathur et al. (2023) for further details about the dataset. We conduct our experiments on the default data split offered in the official dataset release, where the data is segregated into train, test and val in a $0.8 : 0.2 : 0.1$ ratio. All results are reported on the test set.

A.3 Training and Inference

Training During training, the text decoder is fine-tuned to generate the command text, while the mask transformer is fine-tuned for segmentation. The multi-task setup employs a combined weighted loss:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{text}} \cdot \mathcal{L}_{\text{text}} + \lambda_{\text{seg}} \cdot \mathcal{L}_{\text{seg}} \tag{3}$$

The segmentation loss $\mathcal{L}_{\text{seg}}$ is the sum of focal loss Lin et al. (2017) and dice loss Sudre et al. (2017) for the segmentation maps.

Inference During inference, the segmented area is converted into a bounding box. This is achieved by considering points within an $x\%$ radius of the centroid of the mask (with $x = 95$). The contours of the largest contiguous object are then used to determine the coordinates of the bounding box.

A.4 Experimental Set-up

In our experiments, we employed 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 the a cosine scheduler with warm-up approach, incorporating a warm-up period equivalent to 10% of the iterations within each epoch.

In the process of computing loss, we introduced weighting factors, denoted as $\lambda_{\text{text}} = 0.3$ and $\lambda_{\text{seg}} = 1.5$. For segmentation tasks, we employed sigmoid focal loss with parameters $\alpha = 0.25$ and $\gamma = 2$. Furthermore, the decoder incorporated a dropout rate of 0.1.

A.5 BASELINES

Command Generation Baselines

1. **Seq2Seq Text-only baselines** Utilizing GPT2 Radford et al. (2019), BART Lewis et al. (2020), and T5 Raffel et al. (2020), which process only the user text description.
2. **Generator-Extractor:** Incorporating BERT+DETR Devlin et al. (2019); Carion et al. (2020) with an autoregressive decoding head for command generation.
3. **Transformer Encoder-Decoder (Rothe, Narayan, and Severyn 2020):** Combining GPT2 Radford et al. (2019) decoder with LayoutLMv3 Huang et al. (2022) encoder (LayoutLMv3-GPT2) or BERT Devlin et al. (2019) encoder (BERT2GPT2).
4. **Prefix Encoding Mokady et al. (2021):** Using intermediate learned representations from pre-trained encoders (CLIPRadford et al. (2021) and DiT Lewis et al. (2006)) as a prefix to the GPT2 Radford et al. (2019) decoder network and fine-tuning on downstream tasks.
5. **Multimodal Transformer (M4C) Hu et al. (2020):** Combining multimodal input from user description, visual objects, and document text with a text generation decoder instead of the copy pointer mechanism.
6. **DocEditor Mathur et al. (2023):** DocEditor is a task specific baseline that utilises a Transformer-based localization- aware multimodal (textual, spatial, and visual) model. DocEditor decomposes the document image into OCR document content and object boxes, and along with the user request, uses the multimodal transformer to generate the command.
7. **In context learning with LLMs:** We compare our result against GPT3.5 Brown et al. (2020) and GPT4 OpenAI (2023). We use propt based in context learning for these models, specifically providing 3 examples of each command type as context to the model.

Bounding Box Detection Baselines

1. **ReSC-Large Yang et al. (2020):** Method for direct coordinates regression in the RoI bounding box prediction task.
2. **TransVG Deng et al. (2022):** Another approach for direct coordinates regression in the RoI bounding box prediction task.
3. **DocEditor Mathur et al. (2023):** DocEditor encodes the document image by extracting text as OCR and using object detection to capture visual features. The transformer encoded features are used in a Gated R-GCN to generate a layout graph aware representation, which is used downstream to perform bounding box regression.

A.6 LIMITATIONS

1. **Handling of Visual Elements:** The Doc2Command model demonstrates limitations in efficiently executing document editing tasks involving visual elements, such as charts and figures. This challenge stems from the fact that the pretrained backbones are primarily trained on text-dominant document images, leading to suboptimal performance in localizing components within intricate figures. An illustrative example of this limitation is presented in Fig.2(f).

2. **Ambiguity in User Requests:** The model encounters challenges in resolving ambiguity present in user requests. Instances where positional references are either ambiguous or not explicitly specified pose difficulties for the model in accurately interpreting and executing editing commands.
3. **Non-End-to-End System:** It is essential to clarify that Doc2Command does not claim to be an end-to-end document editing system. Instead, it introduces a multitask framework specifically tailored for Region of Interest (RoI) detection and command generation. While excelling in these tasks, the model does not encompass the complete spectrum of functionalities required for comprehensive document editing.
4. **Limitations in Segmentation:** The segmentation process of Doc2Command is subject to certain limitations. The model relies on generating bounding boxes based on the largest continuous object in the segmentation mask. However, in scenarios where the actual region of interest comprises multiple small, patchy masks, the model may struggle to accurately localize the entire region of interest, resulting in patchy segmentation.

Acknowledging these limitations is imperative for a nuanced understanding of Doc2Command's capabilities and areas for potential improvement. Despite these constraints, the model offers a valuable contribution to the domain of document editing, and further research endeavors can address these limitations for enhanced performance.

A.7 Examples

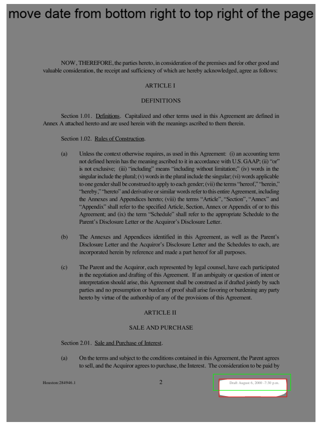
Figure 2 illustrates six instances of our model's performance on the test set. Subfigures (a), (b), and (c) depict correctly inferred examples, while (d), (e), and (f) represent incorrectly inferred examples. Each example within the figure elucidates a distinct capability or limitation of our system.

The instances outlined in Table 3 exhibit six occurrences of commands generated from user requests. Nevertheless, the initial three instances underscore scenarios wherein our model diverges from replicating the ground truth command. Analysis of these errors is given below. In the initial example, although the generated command accomplishes the intended document edit, the ground truth command demonstrates greater efficiency by achieving the same outcome with fewer alterations.The second example shows a delete command, where it differs in descriptiveness from the ground truth command in the initial and final state. In the third example, the model incorrectly considers the action as a modification rather than a replacement, and the state change though expressed differently communicates the intended change by the user. The latter three examples represent correctly predicted edit commands.

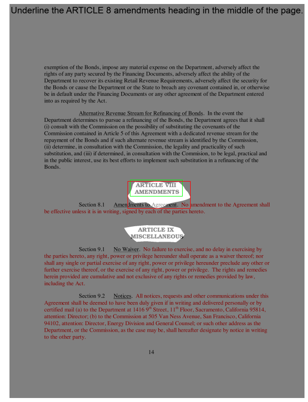
User Request		ACTION_PARA	COMPONENT_PARA	INITIAL_STATE	FINAL_STATE
Change the date "December 1, 2000" to December 11, 2020	*Predicted*	REPLACE	TEXT	December 1, 2000	December, 11, 2000
	Ground Truth	MODIFY	TEXT	1, 2000	11, 2000
Remove all items of "EXPENSE DRIVERS" in table "Exhibit 17. Instinet Group Inc. "	*Predicted*	DELETE	TEXT	present	remove, expense drivers..
	Ground Truth	DELETE	TEXT LIST	present; items in EXPENSE DRIVERS	deleted
change page no 2-32 with roman no ii-xxxii	*Predicted*	MODIFY	TEXT	2-32	ii-xxxii
	Ground Truth	REPLACE	TEXT	numeric	roman
Moved "1. Introduction" from left to mid. Moved page number from mid to left.	*Predicted*	MOVE	TEXT	INTRODUCTION was at left and page number was at mid	INTRODUCTION is at mid and page number is at left
	Ground Truth	MOVE	TEXT	INTRODUCTION was at left and page number was at mid	INTRODUCTION is at mid and page number is at left
Split the last paragraph " Guide providers should jump on this opportunity—" into two paragraphs. New paragraph start with "Guide and search vendors like Yahoo —-"	*Predicted*	SPLIT	PARAGRAPH	not split	split
	Ground Truth	SPLIT	PARAGRAPH	not split	split
added page no. before the number of the page at the centre of footer of the page.	*Predicted*	ADD	TEXT	13	page no. 13
	Ground Truth	ADD	TEXT	13	page no. 13

Table 3: Examples of command generation in Doc2Command. Correct command parameters are highlighted in green, and incorrect command parameters are highlighted in red.

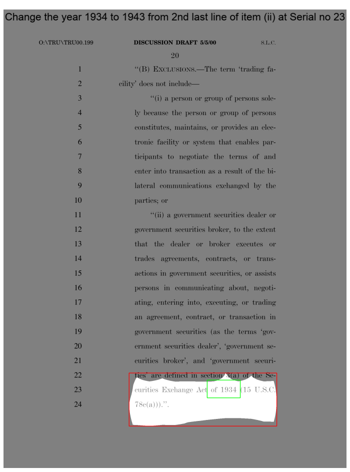
(a) Bounding Box with high IOU: capability to read and recognise text from request in the document.



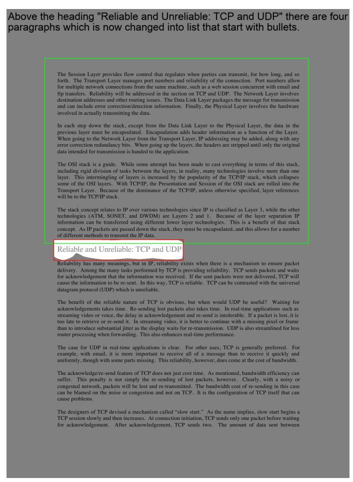
(b) Bounding Box with high IOU: capability to recognise elements such as dates without literal mentions.



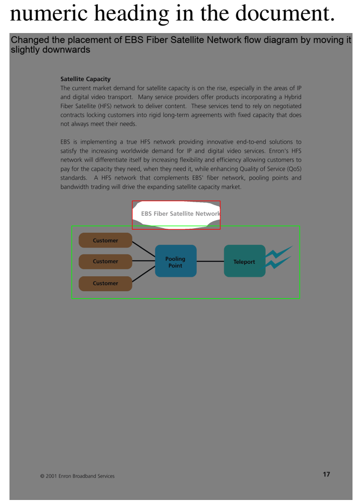
(c) Bounding Box with high IOU: While the command asked to underline "COMMAND 8", Doc2Command was able to semantically identify it with the roman numeric heading in the document.



(d) Bounding Box with low IOU: Doc2Command has localized the request, but is not as specific as the ground truth in this example.



(e) Bounding Box with low IOU: Localizes change to the positional reference here instead of the region of interest.



(f) Bounding Box with low IOU: edit request involves ambiguity between the visual element and its caption.

Figure 2: 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.