DocEdit: Language-Guided Document Editing

Puneet Mathur[1]*, Rajiv Jain[2], Jiuxiang Gu[2],
Franck Dernoncourt[2], Dinesh Manocha[1], Vlad I. Morariu[2]
[1]University of Maryland, College Park
[2]Adobe Research
[1]{puneetm, dmanocha}@umd.edu
[2]{rajijain,jigu,dernoncourt,morariu}@adobe.com

Abstract

Professional document editing tools require a certain level of expertise to perform complex edit operations. To make editing tools accessible to increasingly novice users, we investigate intelligent document assistant systems that can make or suggest edits based on a user's natural language request. Such a system should be able to understand the user's ambiguous requests and contextualize them to the visual cues and textual content found in a document image to edit localized unstructured text and structured layouts. To this end, we propose a new task of language-guided localized document editing, where the user provides a document and an open vocabulary editing request, and the intelligent system produces a command that can be used to automate edits in real-world document editing software. In support of this task, we curate the `DocEdit` dataset, a collection of approximately 28K instances of user edit requests over PDF and design templates along with their corresponding ground truth software executable commands. To our knowledge, this is the first dataset that provides a diverse mix of edit operations with direct and indirect references to the embedded text and visual objects such as paragraphs, lists, tables, etc. We also propose `DocEditor`, a Transformer-based localization-aware multimodal (textual, spatial, and visual) model that performs the new task. The model attends to both document objects and related text contents which may be referred to in a user edit request, generating a multimodal embedding that is used to predict an edit command and associated bounding box localizing it. Our proposed model empirically outperforms other baseline deep learning approaches by 15-18%, providing a strong starting point for future work.

Introduction

Digital documents are used extensively to help people improve business productivity (drafting contract agreements, presentation decks, letterheads, invoices, resumes, form filling) and communicate with customers through online advertisements, social media posts, flyers, posters, billboards, web and mobile app prototypes, etc. However, modern document editing tools require a skilled professional to work on a large screen. Challenges emerge when complex editing operations require multiple different functionalities wrapped within the editing tools for text and image region placement, grouping, spatial alignment, replacement, resizing, splitting, merging, and special effects. As the creation and editing of documents become more ubiquitous and increasingly used by novice users on mobile devices, there is an increasing need to improve the accessibility of these tools through an intelligent assistant system that can understand user's intent and translate it into executable code that can be processed by the editing tool to fulfill a user's editing needs.

We formulate a new task for language-guided document editing and create a new dataset, `DocEdit`, as illustrated in Figure 1, wherein an intelligent system is expected to generate the executable commands (e.g., move components, modify attribute values and special effects, add/delete text, etc.) and visually ground the region of interest given the natural language edit request expressed by human users over a document image. To do so, document editing systems should not only understand the user intent, but also extract and interpret the textual content of the document images along with the visual cues including layout (paragraphs, lists, tables, headers, footers), non-textual elements (marks, tick, shapes, diagrams, graphs), and style (font, colors, size, highlighting, special effects). Departing from generic language-guided image editing tasks (Shi et al. 2020; Lin et al. 2020c), our task warrants a different approach to exploit the above visual cues and high-density of textual tokens by making use of the relative positioning of objects and text tokens.

The new dataset for this task, `DocEdit`, provides natural language edit requests on PDFs and design template documents, along with the result of a human carrying out the edit request. Each edit request is mapped to an executable command that can be simulated in real-world document editing software. To collect such a dataset, we utilized User Interface (UI) experts to, edit a set of input documents, provide a description of the edit, and generate the ground truth executable command corresponding to a set of diverse and creative edit requests posed by freelance designers. `DocEdit` contains more than 17k PDF and 10k design templates with a diverse mix of edit operations (add, delete, modify, split, merge, replace, move, copy) and reference types (direct, object referring, text referring) from the users.

Our work also takes the first step towards automating *Language-guided Localized Document Editing (LLDE)* using `DocEditor`, a Transformer-based localization-aware multimodal (textual, spatial, and visual) model. The model repre-

*Work done during internship at Adobe Research

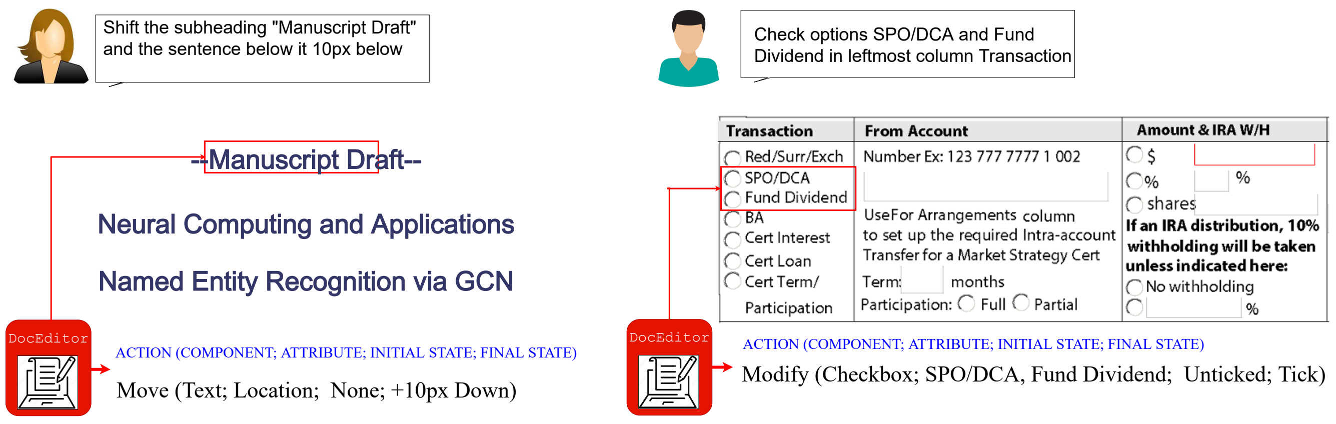
Shift the subheading "Manuscript Draft" and the sentence below it 10px below
--Manuscript Draft--
Neural Computing and Applications
Named Entity Recognition via GCN
DocEditor
ACTION (COMPONENT; ATTRIBUTE; INITIAL STATE; FINAL STATE)
Move (Text; Location; None; +10px Down)
Check options SPO/DCA and Fund Dividend in leftmost column Transaction
Transaction
Red/Surr/Exch
SPO/DCA
Fund Dividend
BA
Cert Interest
Cert Loan
Cert Term/ Participation
From Account
Number Ex: 123 777 7777 1 002
UseFor Arrangements column to set up the required Intra-account Transfer for a Market Strategy Cert
Term: months
Participation: Full Partial
Amount & IRA W/H
$
%
shares
If an IRA distribution, 10% withholding will be taken unless indicated here:
No withholding
%
DocEditor
ACTION (COMPONENT; ATTRIBUTE; INITIAL STATE; FINAL STATE)
Modify (Checkbox; SPO/DCA, Fund Dividend; Unticked; Tick)

Figure 1: The DocEdit dataset provides natural language edit requests on PDFs and design template documents. Each edit request is mapped to an executable command that can be used to automatically apply edits in real-world document editing software. We propose DocEditor, a neural architecture to generate the executable computer command and ground the region of interest bounding box. Examples from the Hierarchical Forms dataset and the public Enron corpus in this figure illustrate several challenges where an intelligent system needs to (a) interpret and localize structured components and their relative positioning in the document; (b) match document text tokens in a text-rich document formatted in varied spatial layouts (checkboxes, choice groups, text fields, columns, rows), (c) visually understand the objects as per the description.

sents the visual appearance of document elements (e.g., paragraphs, images) through their bounding boxes and document semantics (the meaning of the text in the document) through document text tokens obtained via OCR. It uses multi-head attention to obtain a text-enriched visual box embedding which is fused with a text embedding and a regression token. The fused representation is provided to a Transformer decoder, which generates the command text in an autoregressive fashion. Additionally, we employ a layout graph to encode the relative position of boxes and document text tokens to regress the RoI bounding box coordinates. We perform node classification as an auxiliary task for anchor box prediction to ground the edit location in terms of relevant object and document text token boxes. DocEditor proves as a strong benchmark for this task and outperforms other unimodal and multimodal baselines. Our **contributions** are:

- We introduce a new task and dataset for document edit command generation for language-guided localized document editing. The DocEdit dataset consists of document-edit pairs on PDFs and design templates along with corresponding ground truth executable commands.
- We propose DocEditor, a novel multimodal transformer that takes a language-based edit request and produces a spatially localized set of edit commands. To the best of our knowledge, no such multimodal language-guided document editing model exists, with existing models lacking in terms of their understanding of document text and their ability to perform localized edits. Our proposed DocEditor model empirically outperforms other baseline deep learning approaches by 15-18%, providing a strong starting point for future work.

Related Work

Table 1 shows prior tasks and datasets for language-guided image editing systems such as (Shi et al. 2020; Lin et al. 2020c). However, most of these tasks are designed to work with natural images instead of documents that are usually text-rich and may contain a wide range of structured components in varied layouts. Recent GAN-based methods (Jiang et al. 2021a; El-Nouby et al. 2018; Wang et al. 2022; Li et al. 2020; Jiang et al. 2021b) are popular for natural image editing tasks as they perform end-to-end pixel-wise image generation, but are unsuitable for digital-born PDF documents with rich text. Such methods still cannot handle spatial and semantic understanding of embedded text present in the documents. Previous research works (Kim et al. 2022; Shi et al. 2021, 2022; Chen et al. 2018) have explored language-driven image editing to map image edits into actionable computer commands, they are largely limited to global requests where the entire image is uniformly modified. Complex and unstructured documents, for example, images of receipts, invoices, and forms have a large number of relatively small text objects scattered throughout an unstructured document and surrounded by "distraction" objects which are not of interest. Hence, there is a need to spatially localize the objects of interest by modeling text and image content and relating it to the user's text description. Efforts to investigate such

Dataset	Size	OCR	LE	IR	Doc
CAISE	6.1K	✗	✗	✗	✗
DialEdit	8.8K	✗	✗	✗	✗
Edit me	9.1K	✗	✓	✗	✗
ILLC-IER	2.5K	✗	✓	✓	✗
DocEdit (Ours)	28.3K	✓	✓	✓	✓

Table 1: . Comparison of DocEdit with related language-guided image editing datasets. Our dataset is the largest document-centric corpus with localized edits (LE), OCR'ed text, and indirect references (IR) to local objects.

localized editing of spatial regions remain limited. Previous attempts at intent/action/goal identification from user edit requests (Manuvinakurike et al. 2018a,b,c; Lin et al. 2020a) have only explored a limited set of edit functions constrained to changes in brightness, contrast, background color, and their numeric values. Hence, there is a significant gap in the space of operations possible through automated document editing, thus necessitating the development of methods that can generalize to ambiguous open vocabulary user requests and convert them into executable commands grounded in specific action, component, and attribute taxonomy.

Task Description

We introduce the task of generating an executable command from a linguistic user request for editing a document according to the user's intent. Formally, given a document D to be edited and the user request defined as a sequence of n tokens $W = [t_1, t_2, \cdots t_l]$, we predict the executable command C of the format: $ACTION(< Component >, < Attribute >, < Initial\_State >, < Final\_State >, [x, y, h, w])$. Here, *Action* describes the executable function belonging to the following taxonomy - Add, Delete, Copy, Move, Replace, Split, Merge, Modify. It is followed by arguments corresponding to the document *components* to be edited, *attributes* to be modified, *initial state* of the attributes, and the *final state* of the attributes expected in the edited version. The Region of Interest (RoI) is represented by the bounding box $[x, y, h, w]$ enclosing the components to be edited in the input document image, such that (x, y) refers to the top-left coordinate while h and w refer to the height and width of the bounding box, respectively. We perform end-to-end command generation task along with the RoI bounding box regression grounded in the document image.

DocEdit Dataset

Language-guided image editing has been studied in the past. However, there is no existing dataset that captures language-guided editing of structured documents such as PDFs, PowerPoint presentations, and design templates, in which the spatial arrangement of content (text, images, etc) may be as important as the content itself, and edit operations are localized to specific regions of a document. Such documents are rich in layout due to the presence of a high variety of structured components such as tables, graphs, text fields, checkboxes, widgets, lists, and backgrounds along with the unstructured text. Therefore, we present the DocEdit dataset which provides pairs of the document image and user edit requests along with the ground truth edit command and the final edited version of the document. We present two variants of the DocEdit dataset: (1) DocEdit-PDF comprising of edits performed on publicly available PDF documents and (2) DocEdit-Design comprising of edits on design templates.

Data Acquisition: We extracted 20K anonymized PDF documents from the publicly available Enron corpus and Hierarchical forms (Aggarwal et al. 2020) datasets with all personally identifiable information (PII) removed for DocEdit-PDF. We downloaded 12K publicly available and freely distributed design templates from the Adobe Express platform for DocEdit-Design. **Document Edit Creation**: We employed 15 freelance annotators from Upwork with verified past experience in graphic design and Word/PDF document editing. The annotators were provided with examples and online tutorials for editing PDF and design templates and were encouraged to provide creative edit requests unique to each document. The edit requests are shuffled and each annotator is asked to utilize Adobe Acrobat and Adobe Express tools for physically editing PDF documents and design templates, respectively. We trained both sets of annotators to make them familiar with the edit creation process so as to guarantee the quality of the dataset. In the training session, we provided feedback for 100 practice edit requests and the corresponding edited version of the document per annotator for consistency. We performed this training session multiple times until the quality of the data has no obvious/critical issues. **Ground Truth Collection**: We developed a taxonomy of possible actions, components, and attributes. The annotators were asked to select the most relevant edit action along with one or more relevant options for components and attributes. Additionally, we asked the annotators to provide ground truth labels for the initial state of the component prior to editing, and the final state of the component post-editing as text inputs filled in by the annotator based on the user request description and visual context from the document image. In order to uniquely identify the location of the component to be edited, we asked the annotators to mark a tightly enclosed bounding box region surrounding the corresponding component in the document image. We concatenated the labels and bounding box coordinates to form the output command. Ground truth labels were not sourced from the same annotator providing the edit request description. **Data Quality Estimation**: We report a high degree of agreement (Krippendorff's alpha) between annotators for the test portion (20% of the dataset) in the Appendix. **Edit Reference Types**: We further categorize the editing requests as direct, object-referencing, or text-referencing requests. Direct requests are self-contained with specific cues about the component to be modified. We see that a majority of samples in our dataset are indirect requests that refer to a component or text in a document through their relative position, making the task challenging due to the necessity to resolve indirect object references. **Data Splits**: We split both DocEdit-PDF and DocEdit-Design into train, validation, and test in the ratio of 70:10:20.

Methodology

We present DocEditor (see Figure 2), a neural architecture that takes in the user request text and document image as input and predicts an executable edit command by generating the textual functional arguments along with regressing the bounding box coordinates of the edit RoI. Our model can be seen as a sequence of five phases: (a) multimodal feature extraction to obtain the user request embedding, visual object embeddings, and document text token embeddings, (b) obtaining text-enriched visual object representations, (c) generating an executable command by combining the linguistic and visual input representations and passing the combination through a Transformer encoder-decoder, and (d) a layout graph for encoding spatial relationships (e) RoI bounding

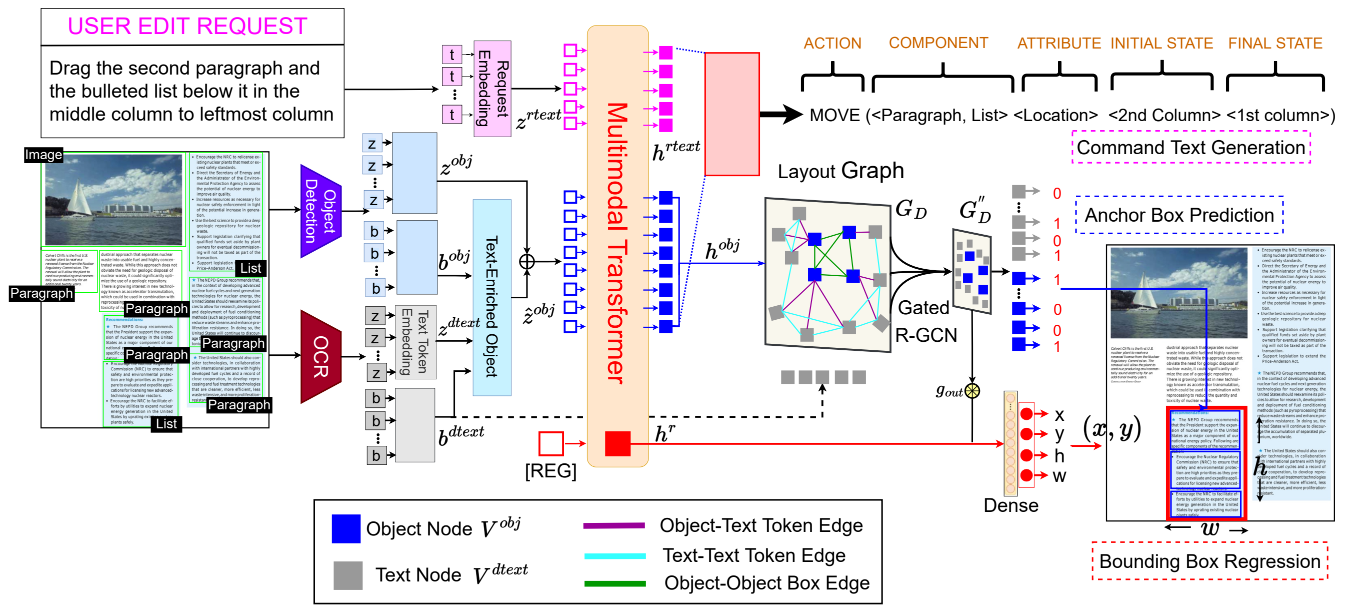

Figure 2: Overview of our proposed system, `DocEditor`: Object boxes and document text tokens obtained by the object detector and OCR system from the document image are combined using multi-headed attention to form text-enriched visual object embeddings. These are concatenated with the encoded text request and [REG] token to form the multimodal input to the Transformer model. A text decoder generates command text in an auto-regressive way. The output hidden states of the object boxes and document text token embeddings are used to create a Layout Graph with nodes joining Object boxes and document text boxes learned through a Gated R-GCN. We perform node classification for anchor box prediction. The graph embedding obtained through the readout function is combined with the [REG] token embedding to regress the RoI bounding box coordinates.

box regression of the command's target region.

`DocEditor` Model

Multimodal Feature Extraction: Our model receives input from three modalities: textual request description, the document's visual objects, and document text tokens. We extract embeddings corresponding to each modality and project them into a common d-dimensional latent space. **(1) Textual Request Embedding**: Given the user request, we encode the request words $w_1, w_2, \cdots, w_i$ into a sequence of T WordPiece tokens using SentencePiece (Kudo and Richardson 2018). We use a vocabulary of 32,000 workpieces obtained from a pre-trained Transformer model to convert the tokens into the request text embedding, and then project them into a d dimensional embedding, yielding $z^{rtext} \in R^{d\times T}$. **(2) Visual Object Embedding**: Given a document image, we use pre-trained object detectors to obtain a set of N visual objects in the document. Inspired by Singh et al. (2019), we extract the visual object features from the object detector's output. These features are linearly transformed into d-dimensional vector space to get the object embedding as $z^{obj} \in R^{d\times N}$. Further, we extract the normalized 2D-bounding box coordinate b_n^{obj} of each object box. **(3) Document Text Embedding**: We obtain a set of M document text tokens from the document image using the OCR system. We extract the 300-dimensional FastText vector (Bojanowski et al. 2017), 604-dimensional Pyramidal Histogram of Characters (PHOC) (Almazán et al. 2014) vector, and normalized 2D bounding box coordinates b_m^{dtext}. We concatenate all the features and linearly project them into a d-dimensional space to get the final document text embedding as $z^{dtext} \in R^{d\times M}$.

Text-enriched Visual Object Representation: Building a common embedding space for user request text, image features, and document text is challenging because there may be hundreds of document text tokens in a text-rich document. Fitting the entire set of document text tokens in the input space may become infeasible due to the increasing computational complexity of multi-headed attention that grows quadratically to the input dimension space. Moreover, not all document text contributes equally to grounding the edit text in the image. There is a need to better exploit the associations between bounding boxes corresponding to document objects (e.g., paragraphs) and the nearby document text at a document level to handle such edit requests that indirectly reference local document objects through their associated document text tokens. Thus, we propose the Text-enriched Document Object representation module (as shown in Fig 2) which contextually integrates the visual objects with their overlapping document text by computing the position-guided attention score vector a_n between the nth visual object and m document text tokens for all $n = 1, \cdots, N$ as follows,

$$a_n = softmax((W^Q b_n^{obj})^T * [W^K b_1^{dtext}, \cdots, W^K b_M^{dtext}] \quad (1)$$

where W^Q and W^K are the query projection matrix and key projection matrix, respectively. The document text attended embedding representation for the nth visual object is calculated as the weighted sum of the M document text embeddings given by following equation $z_n^{obj|dtext} = [z_1^{dtext}, \cdots, z_M^{dtext}] * a_n^T$. Each n^{th} object is then represented by aggregating the object feature embedding z_n^{obj}, document text attended object representation $z_n^{obj|dtext}$ and the linear projection of the object bounding box coordinate $W^{obj}b_n^{obj}$ given as: $\hat{x}_n^{obj} = z_n^{obj} + x_n^{obj|dtext} + W^{obj}b_n^{obj}$. The input sequence of object embeddings is represented by $\hat{z}^{obj} = [\hat{z}_1^{obj}, \cdots, \hat{z}_N^{obj}]$.

Multimodal encoder-decoder for command generation: We first fuse the multimodal input context comprising of user request embedding z^{rtext} and Text-enriched visual object representation $\hat{z}^{obj}$. We further pre-append a learnable embedding (called [REG] token (Deng et al. 2021), and denoted by r) to the multimodal input for mapping the spatial location of the edit intent. The combined multimodal embedding input for our encoder-decoder model is formulated as $z^{input} = z^r \oplus \hat{z}^{obj} \oplus r$, where $\oplus$ represents concatenation. The [REG] token is randomly initialized at the beginning of the training stage and optimized with the whole model.

We then utilize the Text-to-Text (T5) Transformer (Raffel et al. 2020) as our base encoder-decoder architecture to take our input and generate a command sequence. We retain the originally proposed model while modifying the input and output layers to accommodate the additional [REG] token. The multi-head attention mechanism in the Transformer model allows each pair of tokens from the joint embedding to attend to each other across modalities. As a result, the decoder's hidden states as well as the output state of the [REG] token can leverage a consolidated multi-modal representation for localization-aware and layout-oriented command generation and box coordinates regression tasks. The output hidden states from the Transformer model can be represented as $h^{out} = \text{Transformer}(z^{input})$ such that $h^{output} = [h_1^{rtext}, \cdots, h_T^{rtext}; h_1^{obj}, \cdots, h_N^{obj}; h^r]$, where h^{rtext}, h^{obj}, and h^r refer to the output hidden states corresponding to the request text, object, and [REG] embeddings, respectively. We perform greedy decoding, i.e. choose the highest-probability logit at every time step, to generate the output command text.

A Layout Graph to Encode Spatial Relationships: User requests often indirectly reference the components relative to other neighboring objects or text in the document. We hypothesize that the model should reason about the local layout within the region of interest for improving its predictive performance. Hence, we build a Document Layout Graph $G_D = (V, E)$ to encode the relative spatial relations between visual object boxes and text positions. Here, $V = \{V^{obj}, V^{dtext}\}$, where V^{obj}, V^{dtext} are the set of nodes corresponding to N object nodes $V_1^{obj}, \cdots, V_N^{obj}$, and M document text token nodes $V_1^{dtext}, \cdots, V_M^{dtext}$, respectively. The node embeddings of object nodes are extracted from the output hidden states corresponding to the object boxes $h_n^{obj} \forall n \in \{1, \cdots, N\}$. In the case of document text token nodes, we directly use the document text token embedding $z_m^{dtext} \forall m \in \{1, \cdots, M\}$ as the node embedding. The Layout Graph contains three types of edges E: **(1) Object-Text Token Edges**: directed edges for node affiliation if the document text token box lies entirely within the object box. **(2) Text-Text Token Edges**: Connecting all neighboring document text token boxes may lead to dense and isolated components, while joining adjacent tokens in the same line may produce disconnected components. We instead build a β-skeleton of all document text token boxes in the document image with $\beta = 1$ (Kirkpatrick and Radke 1985) since such edges provide a balance between connectivity within a local cluster of document text tokens and ensure that the whole graph is one connected component (Berg et al. 1997). The graph is constructed on peripheral points of the document text token boxes with at most one edge between each pair of boxes. All connections in the β-skeleton graph are added as undirected edges to the layout graph. **(3) Object-Object Box Edges**: directed edges weighted by the type of spatial position between two object boxes in the document. Inspired by Yao et al. (2018), we define ten types of spatial relations – inside, overlap, and 8-way orientations including up, down, left, right, upper-left, upper-right, bottom-left, bottom-right.

We use the Gated Relational Graph Convolution Network (GR-GCN), a gated variant of R-GCN to model our layout graph. GR-GCN is able to learn highly relational data relationships in densely-connected graph networks. The layout graph is passed through two layers of GR-GCN to obtain enriched graph node embeddings $G_D^{''}$.

Bounding Box Prediction: Our proposed model directly infers the bounding box coordinates of the region of interest over the document image. We aggregate the node embeddings corresponding to all object and document text token nodes in $G_D^{''}$ using a summation-based graph readout function (Xu et al. 2018) which is mathematically denoted as $g_{out} = \rho(\sum_{v_i \in G_D^{''}} W_g v_i)$, where W_g is a learnable matrix. We concatenate the output state of [REG] token from the Transformer decoder h^r and the readout output g_{out}, and pass it through a regression block which is implemented as an MLP with a ReLU activated fully-connected layer and a prediction head with four outputs for each bounding box coordinate $b^{'}$ as $b^{'} = ReLU(Dense(h^r \oplus g_{out}))$.

Training `DocEditor`

Command Generation Loss: For generating the textual part of the desired output command, we utilize the pre-trained weights of T5 which were obtained by performing a denoising pre-training task on 750 GB cleaned English text data from the publicly-available Common Crawl web archive. We fine-tune the backbone Transformer architecture using standard maximum likelihood, i.e. using teacher forcing (Williams and Zipser 1989) and a cross-entropy loss between predicted token $t_i^{'}$ and ground truth token t_i as $L_{gen} = -\sum_i t_i \log(t_i^{'})$, where $t_i = 1$ for token predicted correctly.

Bounding Box Regression Loss: To address the problem of scaling effects due to varying sizes of the predicted boxes, we predict normalized bounding box coordinates between 0 and 1000, which are then scaled by the document image dimensions to retrieve original dimensions. We utilize a weighted sum of the scale-invariant generalized IoU loss (GIoU) (Rezatofighi et al. 2019) and the smooth L1 loss

	System	EM (%)	Word Overlap F1	ROUGE-L	Action (%)	Component (%)
Baselines	Generator-Extractor	6.6	0.25	0.22	36.7	8.5
	GPT2	11.6	0.76	0.76	79.7	27.2
	BART	19.7	0.78	0.76	81.2	29.5
	T5	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	8.5	0.25	0.27	44.5	9.34
	DiTCap	23.6	0.81	0.80	82.5	25.5
	Multimodal Transformer	31.6	0.82	0.83	83.1	32.4
Ours	**DocEditor**	**37.6**	**0.87**	**0.86**	**87.6**	**40.7**
Ablation	w/o Text Embedding	6.7	0.15	0.12	6.75	6.5
	w/o Visual Embedding	33.6	0.74	0.75	77.5	36.9
	w/o Layout Graph	32.7	0.75	0.76	82.2	37.5
	w/o Bounding Box Regression Loss	33.6	0.80	0.79	85.2	38.2
	w/o Anchor Box Prediction Loss	35.8	0.84	0.83	84.4	39.5

(a) DocEdit-PDF

System	EM (%)	Word Overlap	ROUGE-L	Action (%)	Component (%)	Attribute (%)
Generator-Extractor	10.1	0.33	0.31	33.4	15.9	14.5
GPT2	16.6	0.78	0.76	76.4	24.5	18.2
BART	19.5	0.79	0.77	77.1	25.1	25.3
T5	20.0	0.80	0.78	77.5	25.8	25.7
BERT2GPT2	6.5	0.31	0.30	36.0	18.6	9.5
LayoutLMv3-GPT2	9.6	0.36	0.34	38.3	20.1	12.6
CLIPCap	9.3	0.24	0.25	19.78	13.6	14.2
DiTCap	18.9	0.79	0.77	77.8	25.4	25.6
Multimodal Transformer	32.8	0.83	0.81	79.5	48.6	35.2
DocEditor	**38.2**	**0.86**	**0.86**	**84.5**	**52.2**	**43.5**
w/o Text Embedding	6.1	0.13	0.11	6.4	6.9	6.5
w/o Visual Embedding	34.0	0.77	0.77	79.5	44.2	37.7
w/o Layout Graph	33.5	0.79	0.77	79.1	46.1	38.3
w/o Bounding Box Regression Loss	34.2	0.83	0.82	82.5	47.1	39.2
w/o Anchor Box Prediction Loss	35.0	0.82	0.78	83.3	49.8	41.7

(b) DocEdit-Design

Table 2: Results comparing the performance of DocEditor for command generation with baselines and ablations on DocEdit-PDF and DocEdit-Design datasets. Bold represents the best-performing model. DocEditor outperforms all baseline methods.

for the standard regression problem. Let $b = (x, y, w, h)$ denote the prediction the normalized ground-truth box as $b' = (x', y', w', h')$. The training objective of our bounding box regression is: $L_{bbox} = L_{smooth-l1}(b, b') + \lambda L_{giou}(b, b')$, where $L_{smooth-l1}$ and L_{giou} are the smooth L1 loss and GIoU loss, respectively. λ is a hyperparameter. **Anchor Box Prediction Loss**: Not all object or document text token boxes are relevant to the edit intent. Hence, the model should have the ability to select the ones that highly overlap with the ground truth RoI. We treat each node in the layout graph as an anchor and perform binary node classification to predict if the object or document text token box lies entirely within the ground truth region of interest (RoI). We optimize the anchor prediction as an auxiliary task through the binary cross-entropy loss as $L_{anchor} = -\sum_{V_i \in G''_D} y_i \log V_i$ where $y_i = 1$ if the object box overlaps with RoI, else 0.

Multitask Training: Command generation, bounding box regression and anchor box prediction tasks are all correlated as they share a common linguistic, spatial and visual latent space, and can reinforce each other. Hence, we use multi-task training to optimize both tasks simultaneously. The final optimization uses a weighted sum of L_{gen}, L_{reg}, L_{anchor} such that total loss $L = \lambda_1 L_{gen} + \lambda_2 L_{reg} + (1 - \lambda_1 - \lambda_2) L_{anchor}$, where the weighting factors λ_1, λ_2 are hyperparameters.

Experiments

Baselines: We compare DocEditor against several unimodal and multimodal baselines for the command generation task:**Seq2seq Text-only**: We use GPT2 (Radford et al. 2019), BART(Lewis et al. 2020), and T5 (Raffel et al. 2020) that input only the user text description. **Generator-Extractor** uses BERT+DETR with an autoregressive decoding head for command generation. **Tranformer Encoder-Decoder** (Rothe, Narayan, and Severyn 2020): Combines GPT2 decoder with LayoutLMv3 encoder (LayoutLMv3-GPT2) or BERT encoder (BERT2GPT2). **Prefix Encoding** (Mokady et al. 2021): We utilize intermediate learned representations from a pre-trained encoder (CLIP (Radford et al. 2021) and DiT (Li et al. 2022)) as a prefix to the GPT2 decoder network and fine-tune on downstream tasks. **Multimodal Trans-**

System	DocEdit-PDF Top-1 Acc (%)	DocEdit-Design Top-1 Acc (%)
ReSC-Large	17.04	15.89
TransVG	25.34	24.89
DocEditor	**36.50**	**34.34**
w/o Text Embedding	3.33	3.25
w/o Visual Embedding	22.45	20.47
w/o Layout Graph	14.48	15.56

Table 3: Results comparing the performance of DocEditor for RoI bounding box regression with baselines and ablations on DocEdit-PDF and DocEdit-Design datasets.

Dataset	GPT2	BART	T5
CAISE (Kim et al. 2022)	60.1	59.5	42.8
ILLC-IER (Lin et al. 2020b)	57.7	55.8	46.9
DocEdit-PDF (Ours)	11.6	19.7	20.4
DocEdit-Design (Ours)	16.6	19.5	20.0

Table 4: Results comparing the difficulty of contemporary language-driven image-editing datasets.

former (M4C) Hu et al. (2020): Combines multimodal input from user description, visual objects, and document text with a text generation decoder instead of the copy pointer mechanism. For the RoI bounding box prediction task, we compare DocEditor against visual grounding methods such as ReSC-Large (Yang et al. 2020) and TransVG (Deng et al. 2022) for direct coordinates regression. **Evaluation Metrics**: We report exact match accuracy (EM %), Word overlap F1and ROUGE-L (Lin 2004). In order to evaluate at a more granular level, we compute the exact match accuracy for actions, components, and attributes. We evaluate bounding box prediction in terms of top-1 accuracy (%) (Jaccard overlap $\geq$ 0.5).

Results

Performance Comparison of command generation: Table 2 compares the performance of DocEditor model against other contemporary baselines on the DocEdit-PDF and DocEdit-Design datasets. Our proposed model achieves significantly better performance across both PDF and design template documents when compared to the text-only and multi-modal baselines used in prior command generation work. We attribute this to DocEditor's ability to localize structured components through Text-enriched object box embeddings and contextualize relevant visual objects and document text tokens through multi-head attention in contrast to text-only approaches that lose these visual cues and prior multi-modal approaches that do not leverage the document structure. Moreover, DocEditor exploits the anchor box prediction loss to determine the mutual importance of each object and document text token box which helps it improve over the multimodal transformer baseline. However, it can also be observed that there is ample room for improvement in both types of document settings. We attribute this to the inherent difficulty of the task and motivate further research by discussing current shortcomings through error analysis. **Performance Comparison of RoI Prediction**: We compare the RoI bounding box prediction performance of baselines with the proposed model in Table 3. We re-purpose scene-text visual grounding baselines for our task due to similarity in the input space. Transformer based TransVG (Deng et al. 2021) model outperforms other competitive baselines as it contextual learns the visual and linguistic information through a common embedding space. Our method further improves this architecture by enhancing the output of [REG] token embedding by output from the layout graph. **Ablation Analysis**: Table 2 and 3 analyze the ablations for each component of the DocEditor. The textual modality of the user request is most critical–removing it yields the random baseline. Removing any other model component does not degrade the performance below this benchmark, which aligns with the fact that the edit command generation task cannot be solved without the edit request descriptions. Removing the Layout Graph severely degrades bounding box regression performance as well as text match accuracy because the model loses the ability to spatially localize the relevant objects and document text tokens. Removing the text-enriched object box embedding significantly affects the consistency of text being generated and the regression box overlap as the model can no longer utilize the document text to match the referred component in the descriptions. **Comparison with contemporary tasks**: We compare the difficulty of our proposed language-guided document editing task with existing image editing tasks through their performance on naive text generation models. We hypothesize that if the text-only modality can provide enough information for solving the task, it will make the image modality redundant and trivialize the overall task to seq2seq generation. Table 4 summarizes the performance of GPT2, BART, and T5 across language-guided image editing datasets - CAISE (Kim et al. 2022) and ILLC-IER (Lin et al. 2020b). We observe that text-only models achieve a high exact match accuracy ($\tilde{6}0\%$). We conclude that samples in these datasets contain many generic edit commands that are neither user-specific nor require a visual or spatial understanding of localized components. Our dataset struggles to achieve one-third of performance ($\leq 20\%$) compared to other datasets, necessitating research in non-trivial multimodal methods for closing the performance gap with expert humans. We present and discuss detailed qualitative examples in the Appendix. We observe that the proposed model is unable to handle commonsense reasoning on world knowledge and makes errors when it is required to parse several attribute modifications simultaneously for the same component.

Conclusion and Future Work

We present a dataset for language-guided document editing with instances of user edit requests on PDFs and design templates and their ground truth executable command for real-world document editing automation. We also present DocEditor, a Transformer-based localization-aware multimodal model that outperforms the competitive baseline for command generation tasks and edit RoI prediction tasks. We provide qualitative analysis with examples to gain insights on the limitations of the proposed model to motivate future work along several interesting directions of conversational document editing and intelligent document assistance.

References

Aggarwal, M.; Gupta, H.; Sarkar, M.; and Krishnamurthy, B. 2020. Form2Seq : A Framework for Higher-Order Form Structure Extraction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 3830–3840. Online: Association for Computational Linguistics.

Almazán, J.; Gordo, A.; Fornés, A.; and Valveny, E. 2014. Word spotting and recognition with embedded attributes. *IEEE transactions on pattern analysis and machine intelligence*, 36(12): 2552–2566.

Berg, M. d.; Kreveld, M. v.; Overmars, M.; and Schwarzkopf, O. 1997. Computational geometry. In *Computational geometry*, 1–17. Springer.

Bojanowski, P.; Grave, E.; Joulin, A.; and Mikolov, T. 2017. Enriching word vectors with subword information. *Transactions of the association for computational linguistics*, 5: 135–146.

Chen, J.; Shen, Y.; Gao, J.; Liu, J.; and Liu, X. 2018. Language-based image editing with recurrent attentive models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8721–8729.

Deng, J.; Yang, Z.; Chen, T.; gang Zhou, W.; and Li, H. 2021. TransVG: End-to-End Visual Grounding with Transformers. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 1749–1759.

Deng, J.; Yang, Z.; Liu, D.; Chen, T.; gang Zhou, W.; Zhang, Y.; Li, H.; and Ouyang, W. 2022. TransVG++: End-to-End Visual Grounding with Language Conditioned Vision Transformer. *ArXiv*, abs/2206.06619.

El-Nouby, A.; Sharma, S.; Schulz, H.; Hjelm, R. D.; Asri, L. E.; Kahou, S. E.; Bengio, Y.; and Taylor, G. W. 2018. Keep Drawing It: Iterative language-based image generation and editing. *ArXiv*, abs/1811.09845.

Hu, R.; Singh, A.; Darrell, T.; and Rohrbach, M. 2020. Iterative answer prediction with pointer-augmented multimodal transformers for textvqa. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9992–10002.

Jiang, W.; Xu, N.; Wang, J.-Y.; Gao, C.; Shi, J.; Lin, Z. L.; and Liu, S. 2021a. Language-Guided Global Image Editing via Cross-Modal Cyclic Mechanism. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2095–2104.

Jiang, Y.; Huang, Z.; Pan, X.; Loy, C. C.; and Liu, Z. 2021b. Talk-to-Edit: Fine-Grained Facial Editing via Dialog. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 13779–13788.

Kim, H.; Kim, D. S.; Yoon, S.; Dernoncourt, F.; Bui, T.; and Bansal, M. 2022. CAISE: Conversational Agent for Image Search and Editing. In *AAAI*.

Kirkpatrick, D. G.; and Radke, J. D. 1985. A framework for computational morphology. In *Machine Intelligence and Pattern Recognition*, volume 2, 217–248. Elsevier.

Kudo, T.; and Richardson, J. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; and Zettlemoyer, L. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *ACL*.

Li, B.; Qi, X.; Lukasiewicz, T.; and Torr, P. H. S. 2020. ManiGAN: Text-Guided Image Manipulation. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 7877–7886.

Li, J.; Xu, Y.; Lv, T.; Cui, L.; Zhang, C.; and Wei, F. 2022. Dit: Self-supervised pre-training for document image transformer. *ACM Multimedia 2022*.

Lin, C.-Y. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, 74–81.

Lin, T.-H.; Bui, T.; Kim, D. S.; and Oh, J. 2020a. A Multimodal Dialogue System for Conversational Image Editing. *ArXiv*, abs/2002.06484.

Lin, T.-H.; Rudnicky, A.; Bui, T.; Kim, D. S.; and Oh, J. 2020b. Adjusting image attributes of localized regions with low-level dialogue. *arXiv preprint arXiv:2002.04678*.

Lin, T.-H.; Rudnicky, A. I.; Bui, T.; Kim, D. S.; and Oh, J. 2020c. Adjusting Image Attributes of Localized Regions with Low-level Dialogue. In *LREC*.

Manuvinakurike, R.; Brixey, J.; Bui, T.; Chang, W.; Artstein, R.; and Georgila, K. 2018a. Dialedit: Annotations for spoken conversational image editing. In *Proceedings 14th Joint ACL-ISO Workshop on Interoperable Semantic Annotation*, 1–9.

Manuvinakurike, R. R.; Brixey, J.; Bui, T.; Chang, W.; Kim, D. S.; Artstein, R.; and Georgila, K. 2018b. Edit me: A Corpus and a Framework for Understanding Natural Language Image Editing. In *LREC*.

Manuvinakurike, R. R.; Bui, T.; Chang, W.; and Georgila, K. 2018c. Conversational Image Editing: Incremental Intent Identification in a New Dialogue Task. In *SIGDIAL Conference*.

Mokady, R.; Hertz, A.; Bermano, A. H.; and . 2021. ClipCap: CLIP Prefix for Image Captioning. *ArXiv*, abs/2111.09734.

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. 2021. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 8748–8763. PMLR.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P. J.; et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140): 1–67.

Rezatofighi, H.; Tsoi, N.; Gwak, J.; Sadeghian, A.; Reid, I.; and Savarese, S. 2019. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 658–666.

Rothe, S.; Narayan, S.; and Severyn, A. 2020. Leveraging Pre-trained Checkpoints for Sequence Generation Tasks. *Transactions of the Association for Computational Linguistics*, 8: 264–280.

Shi, J.; Xu, N.; Bui, T.; Dernoncourt, F.; Wen, Z.; and Xu, C. 2020. A Benchmark and Baseline for Language-Driven Image Editing. In *Proceedings of the Asian Conference on Computer Vision*.

Shi, J.; Xu, N.; Xu, Y.; Bui, T.; Dernoncourt, F.; and Xu, C. 2021. Learning by Planning: Language-Guided Global Image Editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 13590–13599.

Shi, J.; Xu, N.; Zheng, H.; Smith, A.; Luo, J.; and Xu, C. 2022. SpaceEdit: Learning a Unified Editing Space for Open-Domain Image Color Editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 19730–19739.

Singh, A.; Natarajan, V.; Shah, M.; Jiang, Y.; Chen, X.; Batra, D.; Parikh, D.; and Rohrbach, M. 2019. Towards vqa models that can read. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 8317–8326.

Wang, J.; Lu, G.; Xu, H.; Li, Z.; Xu, C.; and Fu, Y. 2022. ManiTrans: Entity-Level Text-Guided Image Manipulation via Token-wise Semantic Alignment and Generation. *ArXiv*, abs/2204.04428.

Williams, R. J.; and Zipser, D. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2): 270–280.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*.

Yang, Z.; Chen, T.; Wang, L.; and Luo, J. 2020. Improving One-stage Visual Grounding by Recursive Sub-query Construction. *ArXiv*, abs/2008.01059.

Yao, T.; Pan, Y.; Li, Y.; and Mei, T. 2018. Exploring visual relationship for image captioning. In *Proceedings of the European conference on computer vision (ECCV)*, 684–699.