Block Diagram-to-Text: Understanding Block Diagram Images by Generating Natural Language Descriptors

Shreyanshu Bhushan[1] and **Minho Lee**[1,2]
[1]Department of Artificial Intelligence, Kyungpook National University, South Korea
[2]ALI Co., Ltd., South Korea
{shreyanshubhushan,mholee}@gmail.com

Abstract

Block diagrams are very popular for representing a workflow or process of a model. Understanding block diagrams by generating summaries can be extremely useful in document summarization. It can also assist people in inferring key insights from block diagrams without requiring a lot of perceptual and cognitive effort. In this paper, we propose a novel task of converting block diagram images into text by presenting a framework called "BloSum". This framework extracts the contextual meaning from the images in the form of triplets that help the language model in summary generation. We also introduce a new dataset for complex computerized block diagrams, explain the dataset preparation process, and later analyze it. Additionally, to showcase the generalization of the model, we test our method with publicly available handwritten block diagram datasets. Our evaluation with different metrics demonstrates the effectiveness of our approach that outperforms other methods and techniques.

1 Introduction

Block diagrams are commonly used to represent a process or workflow of a system, especially the diagrams with different shapes connected with arrows. These types of diagrams are generally found in industry reports, scientific magazines or papers. However, different people use different shapes for a particular notation which makes it quite challenging to understand (Montalvo, 1990).

Block diagram summarization is a task where the goal is to extract the contextual information and relationship between different shapes or nodes from the image, and summarizes the key points in natural language. There are several key benefits and applications of block diagram summarization. First, most of the documents not only contain text but also block diagrams. In order to summarize a document automatically, Artificial Intelligence (AI) needs to understand those block diagrams as well. Automatic generation of description from a block diagram image will lead to better analysis of the related document. Second, descriptive text of a block diagram can be further used for the question and answering (Q&A) task (Kwiatkowski et al., 2019). Third, block diagram summaries can assist individuals to recognize important insights from diagrams that they may have missed otherwise. It is a well-known fact that captions or small descriptions help readers to find important keypoints from the diagrams. It can also help writers to compose effective reports and articles on data facts suggested by automatic explanatory texts. Block diagram summarization offers one more significant advantage of making diagrams more accessible to visually impaired people. With the help of descriptions, they can read using screen readers and understand what is being presented in the block diagram.

Regardless of its various advantages and applications, the block diagram summarization problem has not received much attention in the NLP community. We found no literature regarding block diagram summarization. Early approaches focus mainly only on the detection of different shapes in the diagram (Julca-Aguilar and Hirata, 2018) or converting the handwritten block diagrams to computerized or electronic format (Schäfer and Stuckenschmidt, 2019; Schäfer et al., 2021; Schäfer and Stuckenschmidt, 2021). But none of them consider about relating text phrases with shapes and arrows which plays an important role in summarization tasks. Recently, researchers considered data-driven neural models for describing tabular data (Mei et al., 2016; Gong et al., 2019). Also, few researchers considered chart-to-text for describing different types of chart images (Balaji et al., 2018; Obeid and Hoque, 2020). However, compared to tables and charts, block diagram serves a different problem which consists of lots of variations and complexity. For example, some diagrams contain a single parent and child node whereas some

Findings of the Association for Computational Linguistics: AACL-IJCNLP 2022, pages 153–168
November 20–23, 2022. ©2022 Association for Computational Linguistics

diagrams contain two or more parents or child nodes with different varieties of arrow structures that makes it more complex. There are two main difficulties in addressing the block diagram summarization task. First, the lack of computerized block diagram dataset makes it difficult to solve the task using deep learning models. To our knowledge, there is no dataset available for computerized block diagrams that contain human written summaries. Second, there are no strong baselines for the block diagram summarization task.

In this paper, we present a framework called "BloSum" that converts the block diagram images into text. This framework extracts the contextual meaning and relationships between nodes from the images in the form of triplets <head, relation, tail> which helps the language model in summary generation. Triplets play an important role in data-to-text generation (Gatt and Krahmer, 2018), generally used to represent knowledge graph (KG) (Gardent et al., 2017). Additionally, we present a new dataset for computerized block diagrams (CBD) consisting of 502 diagrams with more than 13,000 annotated elements (shapes, edges, and text phrases) and make our dataset available on GitHub[1]. We introduce three variations of problems mainly based on arrow structure: (i) Break arrows (that have some gap in between an arrow) (ii) Connected arrows (where two or more arrows are interlinked together) (iii) Normal arrows (single arrows including both thin and thick types). These different scenarios motivate us to combine computer vision (CV) and natural language generation (NLG) techniques. Additionally, we test the BloSum with publicly available handwritten block diagram datasets i.e., FC_A (Awal et al., 2011) and FC_B (Bresler et al., 2016) to demonstrate the generalization of the model. For a fair comparison, we extend those two datasets by writing high-quality summaries and triplets. The main contributions of this paper are as follows;

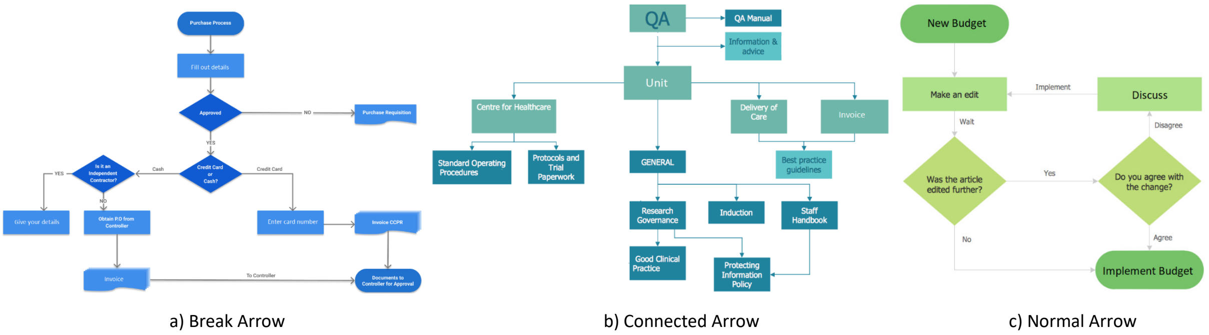
- We propose "BloSum", a new framework for summarization of block diagram images.
- We introduce a new dataset for computerized block diagrams covering a wide range of topics and variations in shape and arrow types.
- We extend the publicly available handwritten block diagram datasets for summarization task.
- We conduct several automatic and human evaluations to check the performance of the proposed model. In addition, the in-depth qualitative analyses uncover some of the key challenges in block diagram summarization.

2 Related Works

Image to Data Generation Earlier, Julca-Aguilar and Hirata (2018) trained the well-known Faster R-CNN object detection pipeline. Standard object-based approaches are unable to identify edges because the arrow bounding boxes are insufficient to identify the relationship between shapes and arrows. To overcome this limitation, Schäfer et al. (2021) added an arrow keypoint predictor to Faster R-CNN. This keypoint predictor predicted the head and tail keypoints of an arrow that helped in finding the relationship between shapes. However, the major downside of this work is that they failed to detect and relate text phrases with shapes and arrows. Moreover, Schäfer and Stuckenschmidt (2021) outperformed the Arrow R-CNN by modeling arrow as a relation between two shapes, and not as standalone objects with bounding boxes. They improved the performance in detecting arrows, but again didn't consider about text phrases relationship with shapes. Our work addresses these issues by considering the text phrase relations for both simple and complex diagrams. Balaji et al. (2018) proposes chart summarization based on a predefined template. A key limitation of template-based work is their limited scalability and flexibility. Moreover, they offer little variation with regard to grammatical styles and lexical choices. In contrast, we focus on the generic block diagram-to-text problem without using any predefined template that contains lots of variations and complexity.

Data to Text Generation Data to text model aims to generate a descriptive text from data or a set of triplets. The task of generating text from data started after the creation of sports summaries from game records (Robin, 1995; Tanaka-Ishii et al., 1998). Recent efforts made use of neural encoder-decoder mechanisms (Puduppully et al., 2019; Kale and Rastogi, 2020; Chen et al., 2020). Pre-trained Language Models (PLMs) such as BERT (Devlin et al., 2019), XLNet (Yang et al., 2019), or RoBERTa (Liu et al., 2019) have established a baseline performance for many natural language understanding (NLU) tasks. However, for many

[1] https://github.com/shreyanshu09/Block-Diagram-Datasets



a) Break Arrow b) Connected Arrow c) Normal Arrow

Figure 1: Sample images for three different categories from our dataset.

NLG tasks, generative PLMs had set a benchmark such as GPT (Brown et al., 2020), BART (Lewis et al., 2020), and T5 (Raffel et al., 2019). T5 model has also state-of-the-art performance on more than twenty natural language processing (NLP) tasks such as GLUE (Wang et al., 2019b), CNN/Daily Mail (See et al., 2017), SuperGLUE (Wang et al., 2019a), SQuAD (Rajpurkar et al., 2018) and many more. It's very uncommon for a single technique to yield consistent advancement across so many tasks. Based on this, we adopt the T5 model in our framework for generating sentences.

Image Captioning Due to the availability of large-scale datasets, there has been quick advancement in image captioning (Agrawal et al., 2019; Chen et al., 2015). Zhang et al. (2021) developed a model to summarize objects from images using an object detection model while Sidorov et al. (2020) generate captions from images by extracting a text with the help of OCR. But images with real-world scenes and objects are totally different from block diagrams. Real-world scenes don't have a very complex relationship between objects whereas block diagrams contain relationships between different nodes that carry both textual and mathematical information. This makes the block diagram-to-text problem different from image captioning.

3 Datasets

Block diagram summarization task uses both object detection and language models, which require a lot of annotated images with high-quality summaries written by humans. To the best of our knowledge, there is no publicly available dataset for computerized block diagrams that satisfies our needs. In this work, we introduce a new dataset CBD for complex computerized block diagrams. We explain all datasets along with the process making of CBD in the next subsections.

Arrow Type	Split	Diagrams	Symbols
Break	Train	56	1496
	Validation	19	528
	Test	19	451
Connected	Train	64	1694
	Validation	22	612
	Test	22	563
Normal	Train	180	4590
	Validation	65	1806
	Test	55	1360

Table 1: Statistics for three different categories of CBD dataset based on arrow types.

3.1 CBD Dataset

Data Collection We collect this dataset through web crawling from different search engines such as Google, Yahoo, Bing, and Naver. We manually choose around 502 images that fit for our work and are publicly available. We remove those images that are either in very poor quality or written in a different language other than English. For each diagram, we download the images in high quality and categorize them into three groups based on the structure of arrow. Figure 1 shows some of the sample images from our dataset for three different categories: Break arrow that has some gap in between an arrow, Connected arrows where two or more arrows are interlinked together, and Normal arrow which includes both thin and thick types of arrows. Table 1 shows some of the statistic of different variations in this dataset based on arrow types. Additional details of the CBD dataset are provided in Appendix A.1.

Data Annotation The annotation for this dataset was challenging as few images miss some of the texts inside the shapes. This missing information makes the overall diagram incomplete. To over-

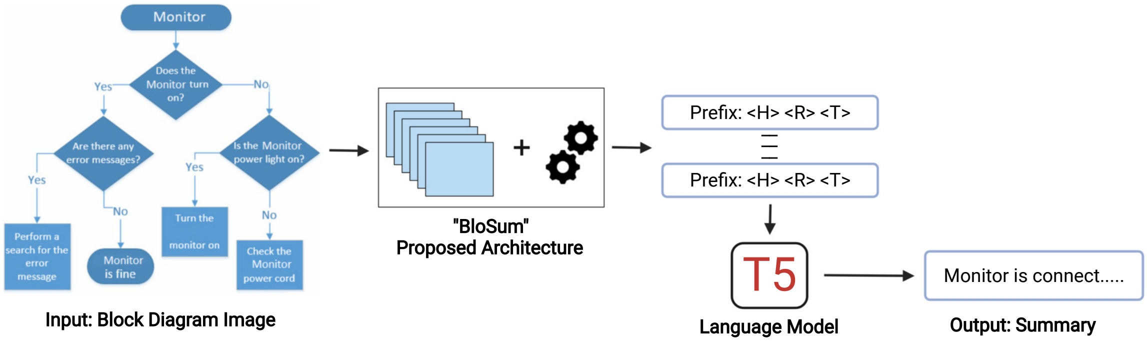

Monitor
Does the Monitor turn on?
Yes
No
Are there any error messages?
Is the Monitor power light on?
Yes
Yes
No
No
Perform a search for the error message
Monitor is fine
Turn the monitor on
Check the Monitor power cord
Input: Block Diagram Image
+
"BloSum" Proposed Architecture
Prefix: <H> <R> <T>
⋮
Prefix: <H> <R> <T>
T5
Language Model
Monitor is connect.....
Output: Summary

Figure 2: Overall architecture of block diagram summarization task.

come this problem, we manually write meaningful texts at those places and then annotate the whole dataset using the LabelImg tool (Tzutalin, 2015). There are total 7 classes: Connection for circle, Data for parallelogram, Decision for diamond, Terminator for eclipse, Arrow, Text, and Process for all other shapes not mentioned above. In this dataset, there are a total of 300 train, 106 validation, and 96 test images that contains more than 13,000 elements (shapes, arrows, and texts). These annotations are helpful for object detection models. However, for the language model, we manually write high-quality summaries along with the triplets in the format of <head, relation, tail> for each diagram.

3.2 Handwritten Block Diagram Dataset

In order to showcase the generalization of our model, we also use two publicly available handwritten block diagram datasets: FC_A (Awal et al., 2011) and FC_B (Bresler et al., 2016). FC_A dataset contains 248 train and 171 test images whereas FC_B contains 280 train, 196 validation, and 196 test images. Diagrams in these datasets are very simple with not many variations and contain only annotated handwritten block diagram images. In order to further use this dataset for the summarization task, we manually write high-quality summaries and triplets for both datasets.

4 Models

In this section, we explain our proposed architecture "BloSum" and all other models used for the block diagram summarization task.

4.1 BloSum

Figure 2 shows the overall architecture of our framework. First, the input image goes into BloSum architecture where it decomposes the images into all possible sets of triplets. This BloSum architecture mainly consists of four parts as shown in Figure 3. We describe each part in detail.

Shape Prediction We consider object detection task for shape prediction to detect all sets of shapes $\boldsymbol{S}$ in an image. For each shape $\mathbf{s} \in \boldsymbol{S}$, it predicts a bounding box $\mathbf{b}_s \in \mathbb{R}^4$ and a class name $\mathbf{c}_s \in \boldsymbol{C}$. Additionally, we set the anchors on each predicted bounding box of shapes at the midpoints of all four sides from where arrows are most likely to be connected. We define $\boldsymbol{C}$ as different classes of shape which include Connection, Data, Decision, Terminator, and Process. Following previous work (Schäfer et al., 2021), we use Faster R-CNN with feature pyramid network (FPN) extension (Lin et al., 2017) but with a different CNN architecture. We use Inception-ResNet-v2 (Szegedy et al., 2017) as a backbone and resize every image to 1024×1024 that we found it suitable in our experiments. We keep an intersection over union (IoU) threshold value of 0.8 for all shape classes and also apply non-maximum suppression (NMS) to eliminate duplicate detections.

Text Prediction We use Faster R-CNN only to predict different shapes. For text and arrow classes, we use different methods because Faster R-CNN shows poor performance in our experiments. We use EasyOCR (Jaided, 2020) for detecting all the sets of text $\boldsymbol{T}$ in an image. It is an open-source tool that works well in detecting texts even from images that contain some noises. For each text $\mathbf{t} \in \boldsymbol{T}$, it predicts a bounding box $\mathbf{b}_t \in \mathbb{R}^4$, confidence score, and the original texts written inside. We combine all the texts $\mathbf{t}$ whose bounding box lies inside the same shape $\mathbf{s}$.

Arrow Prediction Arrow prediction consists of two steps. First, it detects all the arrow lines from

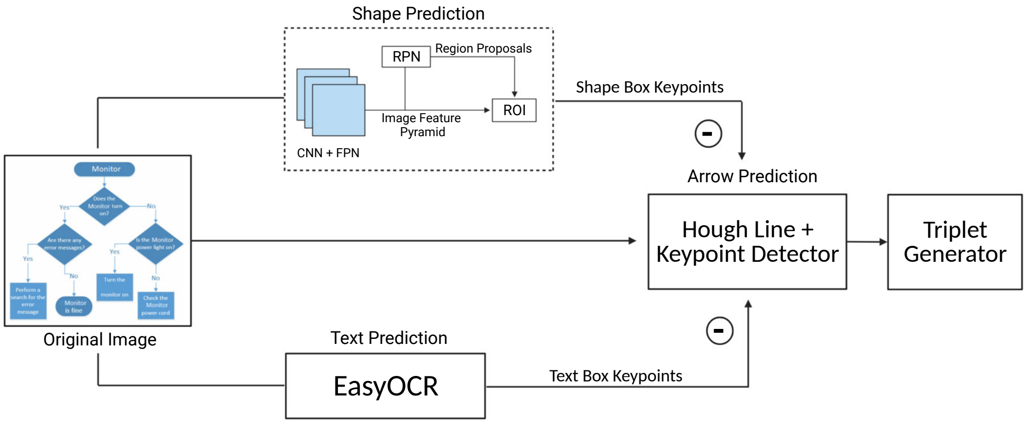


Figure 3: Overall architecture of our proposed method "BloSum".

the diagram including start and end points. Second, it differentiates between head and tail points. Since it is very difficult for any CNN to detect complex arrows such as arrows having a gap or connected arrows. We apply a simple technique in order to detect all sets of arrows $\boldsymbol{A}$ in an image. By using the information from the shape and text prediction, we subtract all the shapes and text phrases from the original image and binarize them. Thus, it remains only with arrows. Then we apply Hough Line Transform in order to detect all the arrow lines and their start and end points. Hough Line Transform helps in detecting the break arrow and the connected arrow as well. To differentiate between the head and tail of an arrow, we add an offset to the start and end points to count the number of white pixels. Finally, a greater number of white pixels represents the head of an arrow, and a lesser number of white pixels represents the tail of an arrow. For each arrow $\mathbf{a} \in \boldsymbol{A}$, predicts 4-d vector $\mathbf{v} = (a^{head}, a^{tail})$ which represents 2-d coordinates of head and tail keypoints per arrow.

Triplet Generator By using all the information from the previous steps, we build a framework called Triplet Generator as shown in Figure 4. This generator finds the connection and relationship between different shapes, and converts these relations into the form of triplets (<H> <R> <T>). For each arrow $\mathbf{a}$ in the diagram, it predicts three things: Head, Relation, and Tail. For each Head and Tail keypoints, first, it finds the closest anchor point placed on shapes. Second, it determines the name of the shape it is associated with. Later, it finds texts inside the shape. It combines all the texts whose bounding boxes lie inside it. If texts are available inside the shape then that particular text is assigned as Head or Tail, and if there are no texts, then the shape name is assigned to Head or Tail. For Relation, first, it determines the distance between the arrow and all the text bounding boxes written outside the shapes. If the distance between arrow and text comes under a threshold value where we set it as 5, then those particular texts are assigned as Relation and if there are no texts which satisfy this condition, then automatically Relation will be assigned as "Connected with". This generator forms a triplet in the top to the bottom and the right to the left order.

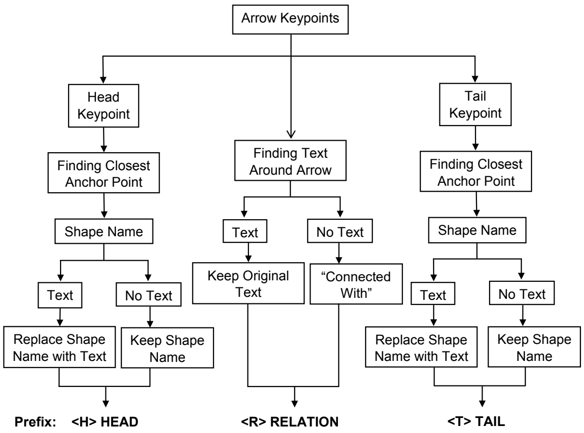


Figure 4: Pipeline of Triplet Generator from BloSum architecture.

After generating all sets of triplets from a diagram, we add "Diagram to Text:" to prefix of each triplet in order to make input friendly for the language model. We experiment with two variants of the T5 model: T5_Large and T5_Base and two variants of the BART model: BART_Large and BART_Base. We also experiment with OCR variants for each model where we replace the extracted text from EasyOCR with their ground truth values. Following previous work (Guo et al., 2020), we connect each token word with an underline "_". For example, "check monitor" is converted to "check_monitor". We use the pre-trained model of each variant on WebNLG 2017 dataset (Gardent et al., 2017). Direct applying these models for our task shows poor performance. Since our dataset contains the ground truth triplets and summaries, we fine-tune each model variant with our dataset.

4.2 Faster R-CNN

We follow the same Faster R-CNN as we use in the BloSum for shape prediction. Instead of detecting only shapes, we predict all the seven classes including text and arrow classes using Inception-ResNet-v2 as a backbone. We keep the IoU threshold value of 0.8 for all classes and also apply NMS. We apply the same EasyOCR for extracting a text from the text bounding box detected by Faster R-CNN. Further for each arrow class, we use the arrow prediction for head and tail keypoints and the triplet generator for generating triplets. Later, those triplets are used by a language model to generate summaries. Similar to BloSum, we experiment with two variants of the T5 model and two variants of the BART model, along with their OCR variants.

4.3 Image Caption

For this category, we consider the Show, Attend, and Tell (SAT) model (Xu et al., 2015) in order to generate captions from block diagram images. We use the pre-trained ResNet50 (He et al., 2016) model on ImageNet (Deng et al., 2009) dataset as the encoder and a unidirectional LSTM (Hochreiter and Schmidhuber, 1997) as the decoder. Since we have the object labels and summaries for the block diagram images, we further fine-tune the model on our dataset. Direct applying without fine-tuning, shows very poor performance for block diagrams.

5 Experiments

5.1 Experimental Setups

All the experiments are done on our machine with 3 GPUs (NVIDIA TITAN RTX) having a memory of 48GB each.

BloSum Julca-Aguilar and Hirata (2018) found that training using the pre-trained model of Faster R-CNN over the MSCOCO dataset (Lin et al., 2014) allows for much faster convergence than training from scratch. Thus, we use the pre-trained model. Although, block diagram images are very different in comparison to the real-world images of the MSCOCO dataset. We then fine-tune the model with our datasets. We use the minibatch training with batch size 1 (due to the variable dimensions of the images) and fix the number of training steps to 25,000. Also, we fix the number of proposals for RPN to 300. Increasing the number of proposals did not result in any considerable improvements.

T5/BART For both language models (T5 and BART), we fine-tune the models with our datasets and use the Adam optimizer (Kingma and Ba, 2015) for maximally 50 epochs with a batch size of 8. The initial learning rate is set to 5×10^{-5}. T5_Large consists of 770M parameters and BART_Large consists of 406M parameters with a 24-layer Transformer as the encoder and decoder whereas T5_Base has 220M parameters and BART_Base has 139M parameters with 12-layer Transformer as the encoder and decoder. For inference, we use the model with the lowest validation loss. Additional training setup of language models are provided in Appendix A.2.

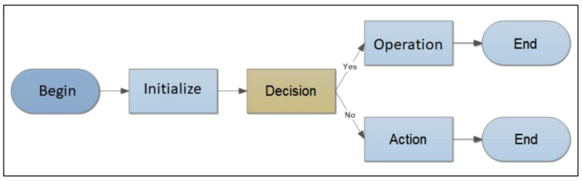
Image Captioning Model We follow the same training setup as presented in the original paper for pretraining both image encoders and captioning model. Run the inference with beam search with a beam size of 4.

5.2 Automatic Evaluation

Measures We conduct automatic evaluation for the generated summaries from different models using five measures. BLEU (Post, 2018) measures how many words in the generated output summaries appeared in the human reference summaries. We use the overall BLEU score obtained by averaging BLEU n-grams (n= 1 to 4) with respect to the brevity penalty. ROUGE-1 (Lin, 2004) measures how many words in the human reference summaries appeared in the generated output summaries. We use the F1 score of ROUGE-1 (Version 1.01) to show the fluency of the sentence generated. BLEURT (Sellam et al., 2020) is a model-based evaluation metric that indicates whether the output sentence is grammatically correct and conveys the correct meaning. We use BLEURT-base-128. Content Selection (CS) metric measures how well the output generated summaries match the ground truth summaries in terms of selecting which records to generate (Wiseman et al., 2017). Finally, we measure Perplexity (PPL) (Radford et al., 2019) using a

Models	CBD					FC_A		FC_B	
	BLEU ↑	ROUGE-1 ↑	CS ↑	BLEURT ↑	PPL ↓	BLEU ↑	ROUGE-1 ↑	BLEU ↑	ROUGE-1 ↑
Image Caption	5.56	10.07	18.42%	-0.84	29.76	3.76	10.9	4.08	13.03
Faster R-CNN + BART_Base	18.01	33.21	40.65%	-0.62	16.84	22.1	44.36	24.67	45.21
Faster R-CNN + BART_Large	17.29	31.16	42.99%	-0.69	17.93	20.07	43.29	22.19	41.63
Faster R-CNN + T5_Base	21.55	38.32	49.43%	0.09	14.66	24.78	**47.52**	24.92	46.35
Faster R-CNN + T5_Large	22.11	40.1	51.78%	0.1	12.06	**25.61**	46.91	**27.81**	**50.47**
BloSum + BART_Base	35.33	75.94	71.64%	0.14	8.44	16.99	34.04	18.16	39.87
BloSum + BART_Large	33.47	75.24	68.16%	0.11	8.33	14.16	31.65	15.49	35.39
BloSum + T5_Base	40.04	78.68	**84.53%**	**0.21**	7.79	19.98	42.34	18.75	40.55
BloSum + T5_Large	**42.18**	**80.78**	83.18%.	0.2	**7.55**	18.23	40.27	20.04	40.85
OCR-Faster R-CNN + T5_Base	28.71	42.92	53.40%	0.13	11.63	48.65	85.79	49.28	85.52
OCR-Faster R-CNN + T5_Large	29.87	45.19	58.05%	0.1	10.91	49.13	86.67	52.45	89.03
OCR-BloSum + T5_Base	40.91	78.74	**84.68%**	**0.21**	7.79	51.01	88.19	52.37	88.92
OCR-BloSum + T5_Large	**42.86**	**81.29**	83.23%.	0.2	**7.54**	**51.73**	**88.24**	**53.17**	**89.56**

Table 2: Automatic evaluation results on computerized (CBD) and handwritten (FC_A, FC_B) datasets from different models. Up arrow ↑ shows, higher is better. Down arrow ↓ shows, lower is better. Bold numbers indicate the best score. "OCR-" models use ground truth OCR values.



Input: Block Diagram Image

BloSum: <H> Begin <R> connected_with <T> Initialize <H> Initialize <R> connected_with <T> Decision <H> Decision <R> Yes <T> Operation <H> Decision <R> No <T> Action <H> Operation <R> connected_with <T> End <H> Action <R> connected_with <T> End

Intermediate result from BloSum model

BloSum+T5_Large: Begin is connected with Initialize which is then connected with Decision. If Decision is Yes then Operation and if Decision is No then Action. Operation and Action both are connected with End.

Final result from our framework

Figure 5: Sample output of a block diagram image from our model.

pre-trained GPT-2 Medium to check the readability and fluency of the generated sentences.

Results Table 2 shows the automatic evaluation results from different models on both computerized and handwritten datasets.

On the CBD dataset, the image caption model fails to extract the relationship between nodes from the diagram and shows a very poor performance while generating descriptions of it. However, language models with Faster R-CNN show a better improvement in extracting relationships between nodes but our proposed method outperforms other models. On one hand, we notice that BloSum with the T5_Large model has the highest BLEU (42.18) and ROUGE-1 (80.78) score. It also generates fluent sentences (low PPL). On the other hand, BloSum with the T5_Base model better captures relevant information from diagrams (high CS score) and grammatically correct sentences (high BLEURT score). But there is a negligible difference as compared to the T5_Large model. Surprisingly, the BART_Base model shows better performance than BART_Large in both Faster R-CNN and BloSum cases. But the low PPL score of the BART_Large model shows that it generates more fluent texts than BART_Base for BloSum. Faster R-CNN mainly fails to detect complex arrows and relations from the diagrams, which results in poor performance of sentence generation. We find similar results for OCR models with negligible improvements for BloSum variants which shows the correctness of text extraction. Overall, BloSum with T5_Large models shows the best performance among others. Figure 5 shows an example of the result obtained by the BloSum model (Intermediate result) and the final result from our framework. The BloSum model produces all the sets of triplets (<H> represents head, <R> represents relation, <T> represents tail) from the given diagram and T5_Large model generates sentences from those triplets.

Also, on the handwritten dataset (FC_A, FC_B), the image caption model shows a very poor performance. Unlike CBD, Faster R-CNN with T5_Large model shows better performance than BloSum. But in the case of OCR models, BloSum with T5_Large models shows the highest BLEU (51.73), ROUGE-1 (88.24) score for FC_A dataset and BLEU (53.17), ROUGE-1 (89.56) score for FC_B dataset. This shows that the BloSum model mainly struggles with handwritten texts, which is because the current version of EasyOCR does not support handwritten texts. Since our work mainly focuses on computerized block diagram images, we left this

	CBD			FC_A			FC_B		
Models	**Adequacy**	**Fluency**	**Coherence**	**Adequacy**	**Fluency**	**Coherence**	**Adequacy**	**Fluency**	**Coherence**
Image Caption	3.6	4.7	4.2	6.4	5.7	5.3	6.8	6.1	5.6
Faster R-CNN +T5_Large	18.6	15.9	13.3	55.6	52.1	50.9	67.4	65.8	65.1
Faster R-CNN +BART_Base	12.7	10.8	11.7	50.3	45.7	43.1	66.1	63.4	62.9
BloSum + T5_Large	68.4	62.3	63.6	28.9	35.6	36.3	36.7	40.2	40.6
BloSum +BART_Base	63.5	60.8	60.9	22.7	28.9	32.1	31.4	38.9	37.5
OCR-Faster R-CNN +T5_Large	30.7	28.2	28.7	60.8	59.4	58.3	64.9	63.2	63.8
OCR-BloSum + T5_Large	**73.3**	**70.1**	**69.8**	**85.4**	**83.1**	**83.9**	**88.8**	**85.1**	**86.4**

Table 3: Human evaluation average score on summaries generated by different models for different datasets. Bold numbers indicate the best score.

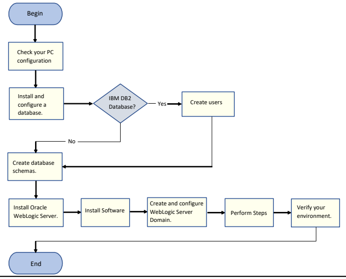
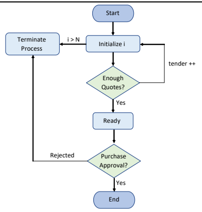
 Begin Check your PC configuration Install and configure a database. IBM DB2 Database? Yes Create users No Create database schemas. Install Oracle WebLogic Server. Install Software Create and configure WebLogic Server Domain. Perform Steps Verify your environment. End	**Faster R-CNN + T5_Large:** Begin is connected with Check your P is connected with Install and configure database which connected with IBM DB2. Create database schemas is connected with Install WebLogic which is connected with Install Software. Install Software is connected with configure WebLogic server connected with Perform Steps. Perform Steps is connected with Verify your environment.	**BloSum + T5_Large:** Begin is connected with Check your PC configuration which is connected with Install and configure a database and further connected with IBM DB2 Database?. If IBM DB2 Database? is Yes then Create users which is connected with Install Oracle WebLogic Server. IBM Database is No then Create database schemas. Install Oracle WebLogic Server is connected with Install Software is connected with Create and configure WebLogic server. Perform Steps is connected with Verify your environment which is finally connected with End of the year.
 Start Initialize i i > N Terminate Process tender ++ Enough Quotes? Yes Ready Rejected Purchase Approval? Yes End	**Faster R-CNN + T5_Large:** Start is connected with Initialize i which is connected with terminate Process. Initialize i is connected with Enough Quotes? Enough Quotes? is Yes then Ready which then connected with Approval? Approval?	**BloSum + T5_Large:** Start is connected with Initialize i which if Initialize i is i>N then terminate Process. Initialize i is connected with Enough Quotes? Enough Quotes? is tender++ then Initialize i and if Enough Quotes? is Yes then Ready which is then connected with Purchase Approval? Purchase Approval? is Rejected then Terminate Process and Purchase Approval? is Yes End.

Figure 6: Sample outputs from CBD dataset from different models (last two columns) along with the block diagram image (first column).

area for the future version of EasyOCR that may support handwritten text as well. Additionally, we test the handwritten datasets by training the model on computerized dataset (CBD) to showcase the generalization of the CBD dataset (Appendix A.3).

5.3 Human Evaluation

Since automatic metrics are only good for small sentences and also no metric is perfect. In our scenario, outputs are long sentences and only humans can perfectly test them. We evaluate the quality of outputs by asking a group of 25 people to rate them based on three quality criteria: (i) Adequacy (whether the sentence clearly expresses the data?); (ii) Fluency (whether the sentences are easy to read and in a natural manner?); (iii) Coherence (whether the sentences are well connected?). For each criterion, people rate on a 0-100 scale where 0 is the "strongly disagree" and 100 is the "strongly agree". We randomly select 40 different block diagram images from each dataset and provide their generated output texts to each examiner.

Table 3 shows the average score given by the examiners. We observe a similar pattern with the automatic evaluation of the performances of different models. For both OCR and non-OCR variants, BloSum with the T5_Large model shows the best performance especially on expressing the data correctly for the CBD dataset. For FC_A and FC_B datasets, the non-OCR BloSum variant fails to detect data correctly mainly because of the non-supporting of handwritten texts by EasyOCR. Faster R-CNN performs well for handwritten texts. However, in OCR variants BloSum with T5_Large model shows the overall best performance in terms of both fluency and coherence. We also determine the mode of the scores given by human evaluators. Details are provided in Appendix A.4.

6 Error Analysis and Challenges

To better analyze the results, we manually choose 50 samples from each dataset obtained by different models as shown in Figure 6. This analysis uncovers some key challenges for vision as well as language tasks that we describe below. Additional sample outputs are provided in Appendix A.5.

Vision Challenge Due to improper detection of some shapes and texts, arrow prediction detects some extra or neglects some pre-existing arrows. This results in the wrong prediction of the triplets, which directly affects the language model in summary generation. Another vision challenge is related to OCR. Block diagrams contain a lot of important information. Since OCRs are not 100% accurate, it detects some wrong texts which lead to error in the facts. More accurate data extraction is necessary for block diagrams.

Imaginary Prediction Imaginary Prediction problem is very common for language models in the data-to-text task. Models sometimes predict some imaginary text which is not relevant to the block diagram image. Some previous works (Wiseman et al., 2017; Parikh et al., 2020) face the same problem for the data-to-text task.

Large Scale Dataset Neural models generally require large-scale datasets. However, our dataset covers a lot of variations but is not big enough. Collecting block diagram images, annotations and their human written summaries are difficult tasks as it requires a lot of manual labor.

7 Conclusion

We have presented a novel task of generating textual descriptions from an image of a block diagram. For this purpose, we propose a new architecture called "BloSum" that extracts the contextual meaning from the diagram in the form of triplets. Additionally, we introduce a new dataset CBD for complex computerized block diagrams with their annotated objects, triplets, and human written summaries. Moreover, for showing the generalization of our model, we tested and extended the publicly available handwritten block diagram datasets i.e., FC_A and FC_B by adding triplets and summaries. This extended dataset can also be used for other data-to-text tasks. Our evaluation with different metrics shows a promising result and outperforms other methods and also reveals some of the unique challenges for this task.

8 Limitations and Future Works

Evaluation with different metrics shows a very promising result of our work. However, there are some limitations such as it does not support electrical diagrams that contain some electrical representations like capacitors, resistors, and others. It only supports those diagrams where shapes are connected through arrows. Also, most of the error occurs in the break arrows category, where there is a very large gap.

To follow up, we plan to explore other approaches to better capture the relationship between shapes, arrows, and texts. We hope that the block diagram summarization task will serve as a useful research for better document summarization as well as for the Q&A task and motivate other researchers to investigate this relatively new area. In future, we also aim to collect more complex diagrams and summaries from different sources and perform experiments to evaluate the generalization of the model.

Ethical Consideration

We had several ethical issues to take into consideration during the dataset collection and preparation process. To respect the intellectual property of the block diagram publishers, we only used publicly available block diagrams that provide publication rights for academic purposes. In addition, we also manually replace around 50% of the text from each diagram with some different meaningful texts. Replacing texts also helps with data privacy issue and protect personal and sensitive data.

The examiners for manual evaluation were randomly selected from the applicants at university. The subjects for this evaluation were those people who wanted to do this evaluation willingly without any wage and have no relation to this project. Additionally, to preserve the privacy of these examiners, all of their evaluations were anonymized.

One potential misuse of our model that we anticipate is the spread of false information. As described in section 6, our model outputs often seem fluent but in reality, they contain certain OCR and imaginary prediction errors. Therefore, these errors could mislead the people if such model outputs are published without being corrected.

Acknowledgements

This work was partly supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2022R1A5A7026673)(50%) and by the NRF grant funded by the Korea government (MSIT) (No. NRF-2021R1A2C3011169)(50%).

References

Harsh Agrawal, Peter Anderson, Karan Desai, Yufei Wang, Xinlei Chen, Rishabh Jain, Mark Johnson, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. nocaps: novel object captioning at scale. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 8947–8956. IEEE.

Ahmad-Montaser Awal, Guihuan Feng, Harold Mouchere, and Christian Viard-Gaudin. 2011. First experiments on a new online handwritten flowchart database. In *Document Recognition and Retrieval XVIII*, volume 7874, pages 81–90. SPIE.

Abhijit Balaji, Thuvaarakkesh Ramanathan, and Venkateshwarlu Sonathi. 2018. Chart-text: A fully automated chart image descriptor. *ArXiv preprint*, abs/1812.10636.

Martin Bresler, Daniel Prusa, and Vaclav Hlavac. 2016. Online recognition of sketched arrow-connected diagrams. *International Journal on Document Analysis and Recognition (IJDAR)*, 19(3):253–267.

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. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Wenhu Chen, Yu Su, Xifeng Yan, and William Yang Wang. 2020. KGPT: Knowledge-grounded pre-training for data-to-text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8635–8648, Online. Association for Computational Linguistics.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. 2015. Microsoft coco captions: Data collection and evaluation server. *ArXiv preprint*, abs/1504.00325.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 248–255. IEEE Computer Society.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The WebNLG challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133, Santiago de Compostela, Spain. Association for Computational Linguistics.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170.

Li Gong, Josep Crego, and Jean Senellart. 2019. Enhanced transformer model for data-to-text generation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 148–156, Hong Kong. Association for Computational Linguistics.

Qipeng Guo, Zhijing Jin, Ning Dai, Xipeng Qiu, Xiangyang Xue, David Wipf, and Zheng Zhang. 2020. \cal p2: A plan-and-pretrain approach for knowledge graph-to-text generation: A plan-and-pretrain approach for knowledge graph-to-text generation. In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+).*, pages 100–106.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

AI Jaided. 2020. Easyocr. *GitHub Repository*.

Frank D Julca-Aguilar and Nina ST Hirata. 2018. Symbol detection in online handwritten graphics using faster r-cnn. In *2018 13th IAPR international workshop on document analysis systems (DAS)*, pages 151–156. IEEE.

Mihir Kale and Abhinav Rastogi. 2020. Text-to-text pre-training for data-to-text tasks. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 97–102, Dublin, Ireland. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

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.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. 2017. Feature pyramid networks for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 936–944. IEEE Computer Society.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv preprint*, abs/1907.11692.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? selective generation using LSTMs with coarse-to-fine alignment. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 720–730, San Diego, California. Association for Computational Linguistics.

Fanya S Montalvo. 1990. Diagram understanding. In *Visual languages and applications*, pages 5–27. Springer.

Jason Obeid and Enamul Hoque. 2020. Chart-to-text: Generating natural language descriptions for charts by adapting the transformer model. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 138–147, Dublin, Ireland. Association for Computational Linguistics.

Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. ToTTo: A controlled table-to-text generation dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186, Online. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-text generation with content selection and planning. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6908–6915. AAAI Press.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *ArXiv preprint*, abs/1910.10683.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Jacques Pierre Robin. 1995. *Revision-based generation of natural language summaries providing historical background: corpus-based analysis, design, implementation and evaluation*. Ph.D. thesis, Columbia University.

Bernhard Schäfer, Margret Keuper, and Heiner Stuckenschmidt. 2021. Arrow r-cnn for handwritten diagram recognition. *International Journal on Document Analysis and Recognition (IJDAR)*, 24(1):3–17.

Bernhard Schäfer and Heiner Stuckenschmidt. 2019. Arrow r-cnn for flowchart recognition. In *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, volume 1, pages 7–13. IEEE.

Bernhard Schäfer and Heiner Stuckenschmidt. 2021. Diagramnet: hand-drawn diagram recognition using visual arrow-relation detection. In *International Conference on Document Analysis and Recognition*, pages 614–630. Springer.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Oleksii Sidorov, Ronghang Hu, Marcus Rohrbach, and Amanpreet Singh. 2020. Textcaps: a dataset for image captioning with reading comprehension. In *European conference on computer vision*, pages 742–758. Springer.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A. Alemi. 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 4278–4284. AAAI Press.

Kumiko Tanaka-Ishii, Koiti Hasida, and Itsuki Noda. 1998. Reactive content selection in the generation of real-time soccer commentary. In *COLING 1998 Volume 2: The 17th International Conference on Computational Linguistics*.

D Tzutalin. 2015. Labelimg. *GitHub Repository*, 6.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019a. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 3261–3275.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019b. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 2048–2057. JMLR.org.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5754–5764.

Pengchuan Zhang, Xiujun Li, Xiaowei Hu, Jianwei Yang, Lei Zhang, Lijuan Wang, Yejin Choi, and Jianfeng Gao. 2021. Vinvl: Making visual representations matter in vision-language models. *ArXiv preprint*, abs/2101.00529.

A Appendices

A.1 Additional Details of CBD Dataset

Figure 7 shows some of the additional statistics of the CBD dataset with respect to shapes, arrows, and text classes. We annotate all the images in PASCAL VOC XML format through the LabelImg tool, which can also be used for other vision tasks. Figure 8 shows some of the complex samples of three categories from our dataset based on arrow structures (Break, Connect, Normal). We write triplets and summaries for all datasets (including handwritten datasets) in text (.txt) format, similar to WebNLG dataset. This format (pair of triplets and summaries) can help other researchers to use it for different data-to-text tasks.

A.2 Additional Details of Language Models

T5 We follow the same training setup as proposed by Guo et al. (2020) and also use their canonicalization of special tokens method in order to handle special tokens. This method converts the special characters that are not in the English alphabet into a format, in which T5 is more familiar. For example, the long dash "—" is converted into a small dash "-". Then each triplet is serialized with special tokens representing the head, relation, and tail. For proper readability, the input format of text phrases such as "check monitor" is actually "check@@ _@@ monitor" because T5 uses byte-pair encoding. For T5, we use two variants: the T5_Large model which consists of 24 attention modules and 770M parameters, and the T5_Base model which consists of 12 attention modules and 220M parameters.

BART For BART, we follow the same training setup as presented by Lewis et al. (2020). It is particularly pre-trained for text generation tasks. Same as T5, we use two variants of BART: i) BART_Large and, ii) BART_Base. Bart-large model consists of 24-layer, 1024-hidden, 16-heads, and, 406M parameters whereas the Bart-base model consists of 12-layer, 768-hidden, 16-heads, and 139M parameters.

A.3 Additional Results from Dataset Evaluation

We additionally perform an experiment with the CBD dataset. First, we train and test the faster R-CNN model on the handwritten datasets (Train: FC_A/FC_B, Test: FC_A/FC_B). Second, we train the same model with CBD and then test it with handwritten datasets (Train: CBD, Test: FC_A/FC_B). Table 4 shows the precision score values of all the seven classes along with the average values. We set the IOU threshold value of 0.7 for all seven classes. Surprisingly, the model detects better handwritten diagrams, when trained on computerized diagrams than trained on handwritten diagrams. However, the CBD dataset does not contain any handwritten diagrams. This shows the generalization and usefulness of our dataset, which can also be used in many other applications.

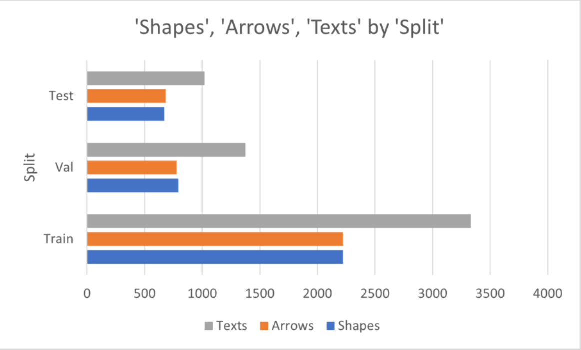


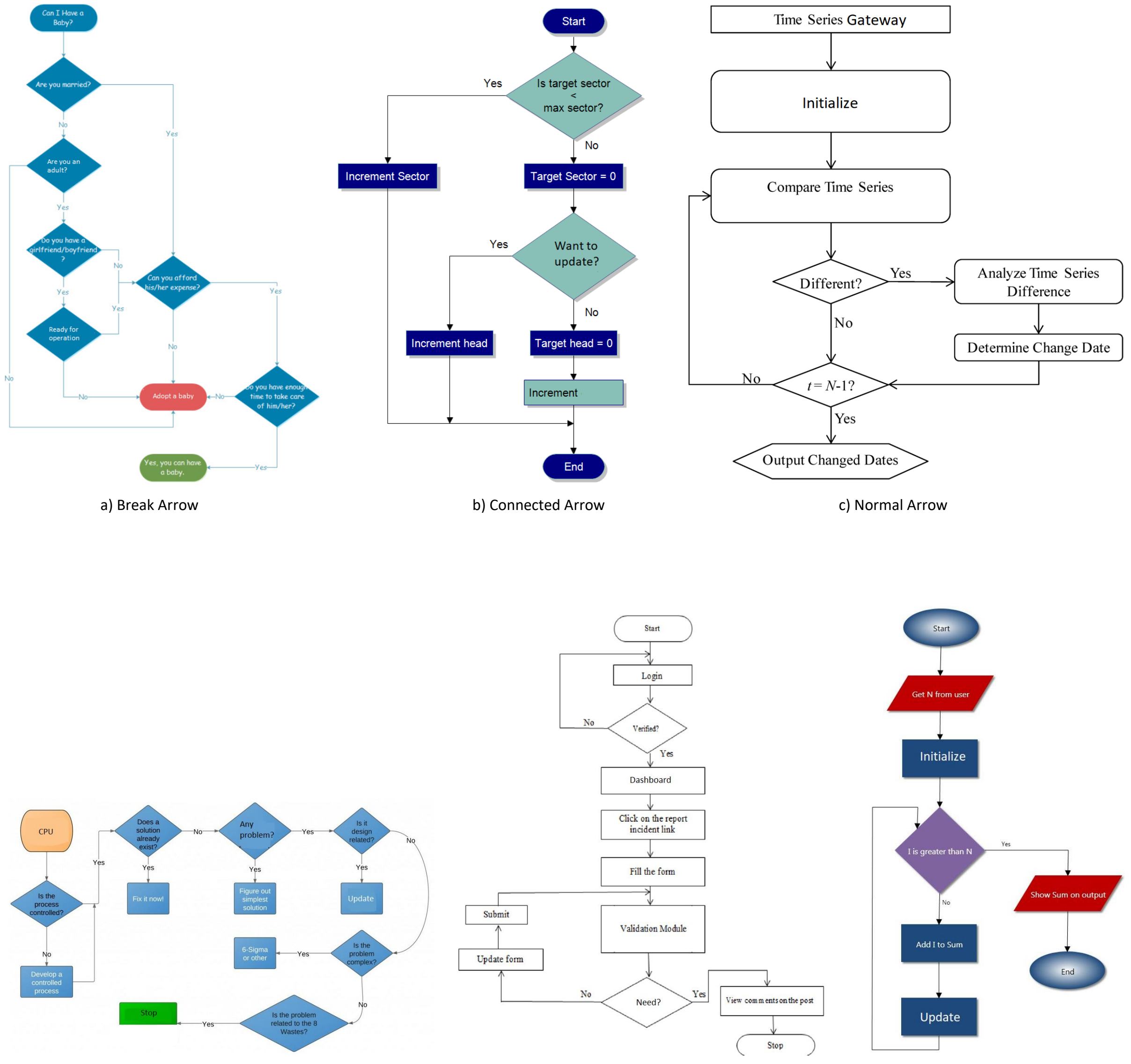
Figure 7: Additional statistics of CBD dataset.

A.4 Additional Results from Human Evaluation

Table 5 shows the mode scores of human evaluation on summaries generated by different models for the different datasets. Mode scores provide some additional insights on the evaluation of the output generated.

A.5 Additional Sample Outputs from CBD, FC_A, and FC_B datasets

Figure 9 shows some of the sample outputs (triplets) generated from our model (BloSum) for the computerized (CBD) dataset. Figure 10 shows some of the additional sample outputs (summaries) generated from our model (BloSum) plus language model for both computerized (CBD) as well as handwritten datasets (FC_A, FC_B).



a) Break Arrow

b) Connected Arrow

c) Normal Arrow

d) Break Arrow

e) Connected Arrow

f) Normal Arrow

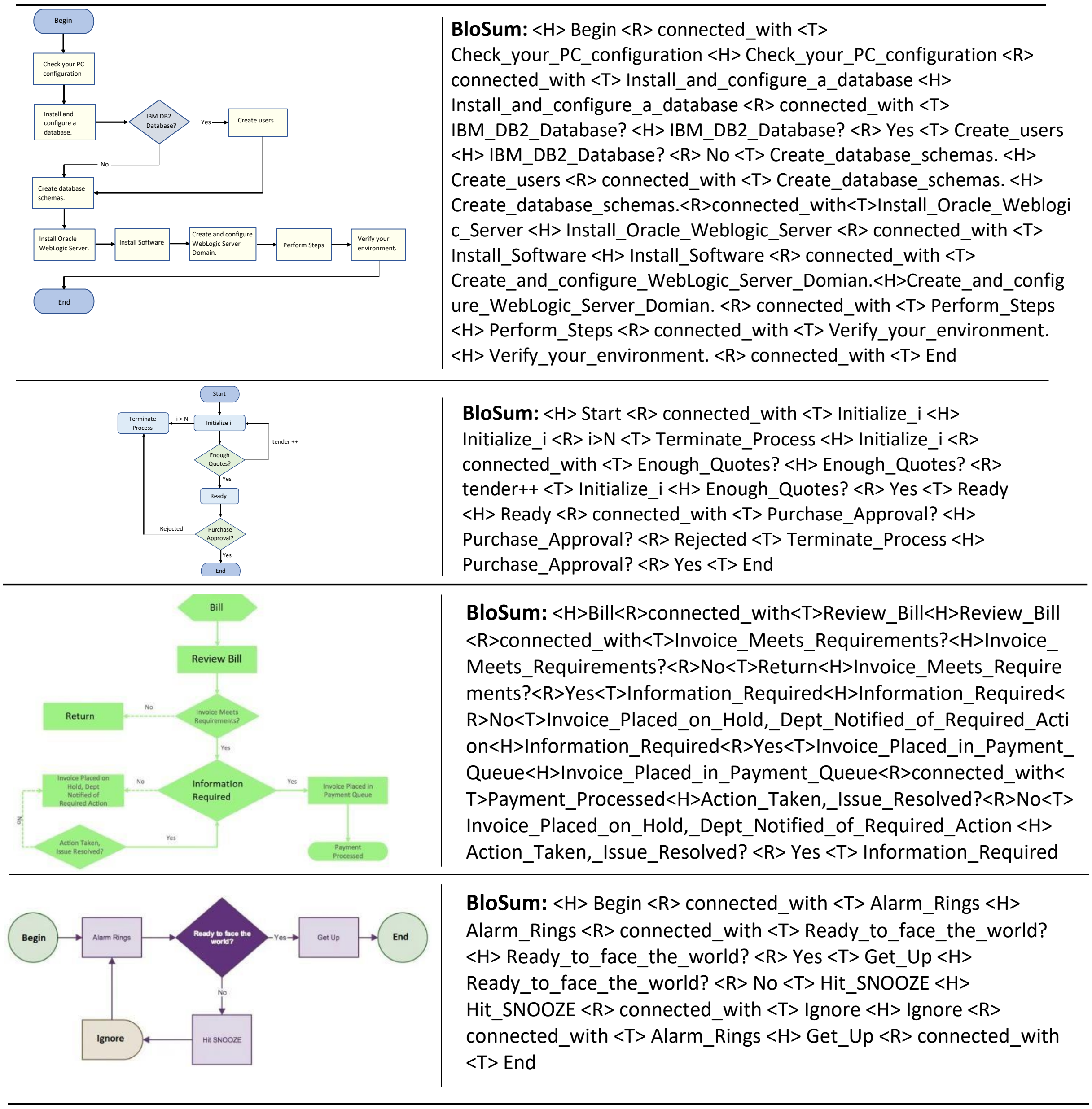
Figure 8: Sample images from CBD datasets for three arrow variations.

Class	Train: FCA Test: FCA	Train: CBD Test: FCA	Train: FCB Test: FCB	Train: CBD Test: FCB
Arrow	86.17	89.65	88.13	90.76
Connection	96.59	99.76	99.44	99.54
Data	99.97	99.99	99.04	99.3
Decision	99.57	99.99	99.99	99.98
Process	99.37	99.55	98.9	99.32
Terminator	99.99	99.83	99.85	99.97
Text	83.13	84.97	86.74	87.04
Average	94.97	**96.24**	96.01	**96.55**

Table 4: Precision score values for different classes on different train and test datasets. Bold numbers indicate the best score.

	CBD			FC_A			FC_B		
Models	**Adequacy**	**Fluency**	**Coherence**	**Adequacy**	**Fluency**	**Coherence**	**Adequacy**	**Fluency**	**Coherence**
Image Caption	3	4	3	5	5	4	6	5	4
Faster R-CNN +T5_Large	15	12	10	50	45	45	60	55	55
Faster R-CNN +BART_Base	10	8	8	40	35	35	60	55	55
BloSum + T5_Large	55	50	50	20	25	30	30	35	35
BloSum +BART_Base	55	50	50	10	15	25	25	30	30
OCR-Faster R-CNN +T5_Large	20	20	20	50	50	50	55	55	50
OCR-BloSum + T5_Large	**60**	**60**	**50**	**75**	**70**	**70**	**75**	**70**	**75**

Table 5: Human evaluation mode score on summaries generated by different models for different datasets. Bold numbers indicate the best score.



BloSum: <H> Begin <R> connected_with <T> Check_your_PC_configuration <H> Check_your_PC_configuration <R> connected_with <T> Install_and_configure_a_database <H> Install_and_configure_a_database <R> connected_with <T> IBM_DB2_Database? <H> IBM_DB2_Database? <R> Yes <T> Create_users <H> IBM_DB2_Database? <R> No <T> Create_database_schemas. <H> Create_users <R> connected_with <T> Create_database_schemas. <H> Create_database_schemas.<R>connected_with<T>Install_Oracle_Weblogic_Server <H> Install_Oracle_Weblogic_Server <R> connected_with <T> Install_Software <H> Install_Software <R> connected_with <T> Create_and_configure_WebLogic_Server_Domian.<H>Create_and_configure_WebLogic_Server_Domian. <R> connected_with <T> Perform_Steps <H> Perform_Steps <R> connected_with <T> Verify_your_environment. <H> Verify_your_environment. <R> connected_with <T> End

BloSum: <H> Start <R> connected_with <T> Initialize_i <H> Initialize_i <R> i>N <T> Terminate_Process <H> Initialize_i <R> connected_with <T> Enough_Quotes? <H> Enough_Quotes? <R> tender++ <T> Initialize_i <H> Enough_Quotes? <R> Yes <T> Ready <H> Ready <R> connected_with <T> Purchase_Approval? <H> Purchase_Approval? <R> Rejected <T> Terminate_Process <H> Purchase_Approval? <R> Yes <T> End

BloSum: <H>Bill<R>connected_with<T>Review_Bill<H>Review_Bill <R>connected_with<T>Invoice_Meets_Requirements?<H>Invoice_Meets_Requirements?<R>No<T>Return<H>Invoice_Meets_Requirements?<R>Yes<T>Information_Required<H>Information_Required<R>No<T>Invoice_Placed_on_Hold,_Dept_Notified_of_Required_Action<H>Information_Required<R>Yes<T>Invoice_Placed_in_Payment_Queue<H>Invoice_Placed_in_Payment_Queue<R>connected_with<T>Payment_Processed<H>Action_Taken,_Issue_Resolved?<R>No<T> Invoice_Placed_on_Hold,_Dept_Notified_of_Required_Action <H> Action_Taken,_Issue_Resolved? <R> Yes <T> Information_Required

BloSum: <H> Begin <R> connected_with <T> Alarm_Rings <H> Alarm_Rings <R> connected_with <T> Ready_to_face_the_world? <H> Ready_to_face_the_world? <R> Yes <T> Get_Up <H> Ready_to_face_the_world? <R> No <T> Hit_SNOOZE <H> Hit_SNOOZE <R> connected_with <T> Ignore <H> Ignore <R> connected_with <T> Alarm_Rings <H> Get_Up <R> connected_with <T> End

Figure 9: BloSum outputs (triplets) from the CBD dataset.

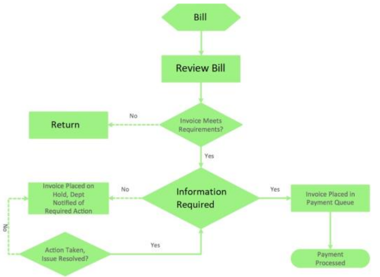
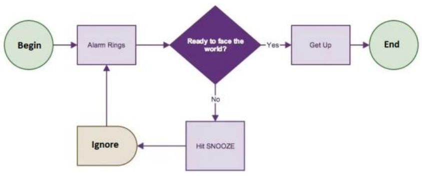
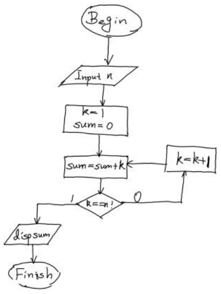
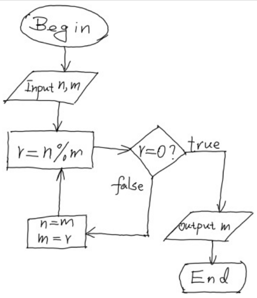
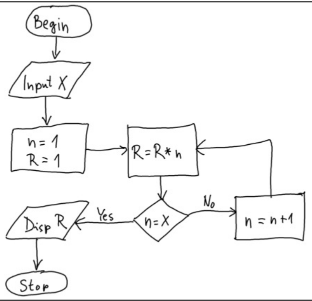
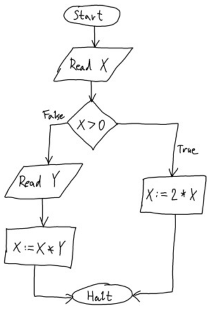
 Bill Review Bill Return No Invoice Meets Requirements? Yes Invoice Placed on Hold, Dept Notified of Required Action No Information Required Yes Invoice Placed in Payment Queue No Action Taken, Issue Resolved? Yes Payment Processed	**BloSum + T5_Large:** Bill is connected with Review Bill which is then connected with Invoice Meets Requirements?. Invoice Meets Requirements? is No then Return and if Invoice Meets Requirements? is yes then Information Required. Information Required is No then Invoice Placed on Hold, Dept Notified of Required Action and if Information Required is Yes then Invoice Placed in Payment Queue which is then connected with Payment Processes. Action Taken, Issue Resolved is No then Invoice Placed on Hold, Dept Notified of Required Action and if Action Taken, Issue Resolved is Yes then Information Required.
 Begin Alarm Rings Ready to face the world? Yes Get Up End No Ignore Hit SNOOZE	**BloSum + T5_Large:** Begin is connected with Alarm Rings which is connected with Ready to face the world?. If Ready to face the world? is Yes then Get Up and it is connected with End. Ready to face the world? is No then Hit SNOOZE which is then connected with Ignore and finally connected with Alarm Rings.
 Begin Input n k=1 sum=0 Sum=sum+k k=k+1 1 k==n? 0 disp sum Finish	**OCR-BloSum + T5_Large:** Begin is connected with Input n and is then connected with k=1 and Sum=0 which is further connected with Sum=Sum+k. k=k+1 is connected with Sum=Sum+k. Sum=Sum+k is connected with K==n?. K==n? is 0 then k=k+1 and if K==n is 1 then disp Sum and then disp sum is connected with Finish.
 Begin Input n,m r=n%m r=0? true false n=m m=r output m End	**OCR-BloSum + T5_Large:** Begin is connected with Input n,m which is then connected with r=n%m. r=n%m is connected with r=0? which if true, then output m and if r=0? is false then n=m and m=r and is connected with r=n%m. Output m is connected with End.
 Begin Input X n=1 R=1 R=R*n No n=n+1 Yes n=X Disp R Stop	**OCR-BloSum + T5_Large:** Begin is connected with Input X and is then connected with n=1 and R=1 which is further connected with R=R*n. R=R*n is connected with n=X. n=n+1 is connected with R=R*n and if n=x is No then n=n+1 and if yes then Disp R. Disp R is connected with Stop.
 Start Read X False X>0 True Read Y X:=2*X X:=X*Y Halt	**OCR-BloSum + T5_Large:** Start is connected with Read X which is then connected with X>0. If X>0 is False then Read Y and if X>0 is True then X=2*X which is further connected with Halt. Read Y is connected with X=X*Y and is connected with Halt.

Figure 10: Sample outputs from CBD (first two rows), FC_A (third and fourth rows), and FC_B (last two rows) datasets.