Towards Table-to-Text Generation with Numerical Reasoning

Lya Hulliyyatus Suadaa[1], **Hidetaka Kamigaito**[1], **Kotaro Funakoshi**[1], **Manabu Okumura**[1] **and Hiroya Takamura**[1,2]
[1]Tokyo Institute of Technology
[2]National Institute of Advanced Industrial Science and Technology (AIST)
lya@stis.ac.id
{kamigaito,funakoshi,oku}@lr.pi.titech.ac.jp
takamura@pi.titech.ac.jp

Abstract

Recent neural text generation models have shown significant improvement in generating descriptive text from structured data such as table formats. One of the remaining important challenges is generating more analytical descriptions that can be inferred from facts in a data source. The use of a template-based generator and a pointer-generator is among the potential alternatives for table-to-text generators. In this paper, we propose a framework consisting of a pre-trained model and a copy mechanism. The pre-trained models are fine-tuned to produce fluent text that is enriched with numerical reasoning. However, it still lacks fidelity to the table contents. The copy mechanism is incorporated in the fine-tuning step by using general placeholders to avoid producing hallucinated phrases that are not supported by a table while preserving high fluency. In summary, our contributions are (1) a new dataset for numerical table-to-text generation using pairs of a table and a paragraph of a table description with richer inference from scientific papers, and (2) a table-to-text generation framework enriched with numerical reasoning.

1 Introduction

Recent data-to-text generation studies have shown significant improvement in generating faithful text aligned with data sources. A copy mechanism has been widely explored to improve faithfulness in various ways. Wiseman et al. (2017) used joint probabilities to let models choose between copying records from data sources or generating from a vocabulary. Puduppully et al. (2019) improved a similar approach by modeling entity representations as a unit of copying. This approach has proven to be effective in generating descriptive text that explicitly mentions facts from sources.

However, as introduced by Chen et al. (2020a), humans have the ability to produce more analytical text with richer inference, including numerical reasoning. Making inferences beyond texts is still an open question due to the limitation of language models in handling numeric operations. In this study, we further encourage research by elaborating numerical tables to initialize the ability to inject reasoning while maintaining high fluency.

Table 2: The overall mention detection results on the test set of OntoNotes.

Model	Precision	Recall	F1
Our full model	89.6	82.2	85.7
Lee et al. (2018)	86.2	83.7	84.9

Target Header
Our full model
Description
Table 2 shows the mention detection results on the test set. Similar to coreference linking results, **our model achieves higher precision and F1 score**, which indicates that our model can significantly reduce false positive mentions while it can still find a reasonable number of mentions.

Figure 1: Example of table and description in numeric NLG dataset.

Our contributions are summarized as follows.

- We introduce a new dataset for table-to-text generation focusing on numerical reasoning. The dataset consists of textual descriptions of numerical tables from scientific papers. Our dataset is publicly available on https://github.com/titech-nlp/numeric-nlg.
- We adopt template-guided text generation (Kale and Rastogi, 2020a) for a table-to-text generation task and propose injecting pre-executed numerical operations in the template to guide numerical-reasoning-based text generation. We compare different types of templates for table representations in pre-trained models.

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 1451–1465
August 1–6, 2021.

- We propose a copy mechanism for pre-trained models, that uses general placeholders covering table contents and results of pre-executed numerical operations to avoid fact hallucination.

- We conduct experiments with current state-of-the-art neural generation models and a simple template-based system to demonstrate the challenges and opportunities for future research on text generation with numerical reasoning.

2 Related Work

The power of tables in presenting data efficiently further encourages research done by exploring the tables as data sources in natural language tasks, such as table-to-text generation (Liang et al., 2009; Wiseman et al., 2017; Lebret et al., 2016; Parikh et al., 2020), table question answering (Pasupat and Liang, 2015; Wang et al., 2018), and table-based fact verification (Chen et al., 2020b; Gupta et al., 2020). Recent research on the table-to-text generation task is starting to generate text with more reasoning. Murakami et al. (2017) explored stock prices to generate market comments by adding generalization tags of possible arithmetic operations to cover mathematical reasoning. Nie et al. (2018) proposed operation-guided attentions by exploring the results of pre-executed numerical operations. The dataset closest to ours is LOGICNLG, by Chen et al. (2020a), who first introduced logical text generation using open-domain tables with unknown schemas. Different from our target text for generation, which consists of several sentences in a paragraph, they proposed a task of generating only one sentence from selected table contents.

3 Numerical Table-to-Text Dataset

We created numericNLG, a new table-to-text dataset focusing on a text generation task with numerical reasoning. We collected table descriptions from scientific papers, that are naturally produced by experts with richer inference.

3.1 Dataset Creation

Data Acquisition We constructed a table-to-text dataset based on numerical tables of experimental results, extracted from PDF files of scientific papers on the ACL Anthology website,[1] introduced by Suadaa et al. (2021). Then, we collected candidates for corresponding descriptions from the source files using PDFMiner.[2] We used table numbers in their captions as keywords for the collection. An example of a table and its description is shown in Figure 1.

Data Cleansing and Annotation Extracted table descriptions can be noisy since they may contain only table numbers without any sentences describing table facts. We hired experts in the computer science field to clean and annotate the extracted descriptions in the following steps:

- Examine tables and their corresponding descriptions and then recommend only the descriptions that have at least one sentence representing numerical facts in the table.

- Categorize each sentence of the recommended description into three fact-checking classes: data description, supporting description, and not-related-to-table description. As a final dataset, we used only sentences classified as belonging to the data description category to reduce fact hallucination.

- Identify a content plan of table description by selecting part of table headers which directly stated or logically inferred in the description, called target header. For example, refer to the table description shown in Figure 1, "Our full model" is selected as the target header.

We used the same split of training, validation, and test sets as the source table dataset (Suadaa et al., 2021).

3.2 Dataset Comparison

Table 1 provides a comparison of numericNLG with other related table-to-text datasets. The ROTOWIRE (Wiseman et al., 2017) dataset consists of summaries of NBA basketball games containing several paragraphs, paired with their corresponding box-score tables. Since ROTOWIRE has only 39 record types, each table contains similar record types with limited schemas. Although most of the ROTOWIRE table contents are in numerical values, the summaries contain only a few numerical-reasoning sentences, such as a comparison of scores between two basketball teams. While our dataset consists of closed domain articles as

[1] https://www.aclweb.org/anthology/

[2] http://pypi.python.org/pypi/pdfminer/

	Tables	Examples	Unit of Desc.	Vocab.	Token/Desc.	Domain	Inference	Schema
ROTOWIRE	4.9K	4.9K	Document	11.3K	337	Sport	Few	Known
LOGICNLG	7.3K	37.0K	Sentence	122.0K	11	Open	Rich	Unlimited
numericNLG	1.3K	1.3K	Paragraph	19.6K	94	Scientific	Rich	Unlimited

Table 1: Dataset comparison.

with ROTOWIRE, it is of shorter text (a paragraph) and with unlimited table schemas.

Chen et al. (2020a) introduced the LOGICNLG dataset to facilitate the study of table-to-text generation tasks with richer inference. The dataset contains unlimited schemas of open-domain tables crawled from Wikipedia, paired with five annotated sentences covering different logical inferences. Although most inferences are numerical reasoning, the table contents are not fully numeric.

Similar in motivation to LOGICNLG in generating text that can be logically entailed by facts in tables, numericNLG consists of collections of paragraphs that are naturally produced by human experts in scientific papers, paired with their corresponding numerical tables. Our dataset has fewer tables than LOGICNLG, focusing on numerical-reasoning text in the scientific domain.

4 Table Representation

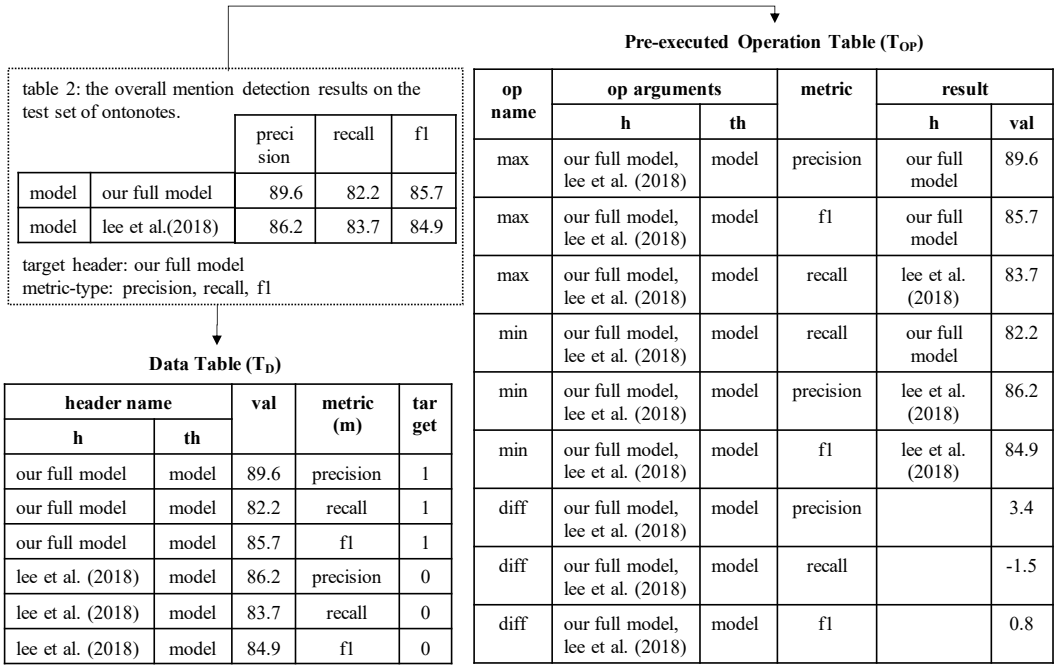
Due to ROTOWIRE's limited schemas, Wiseman et al. (2017) viewed a table input as a set of records (entity, value, type), where the entity and the type are the extracted row and column names, respectively. Because of the unlimited table schemas in our dataset, by capturing the original table structure in real-world tables, this paper uses the representations which consist of captions, row headers, column headers, cell values, and metrics, called a data table. Using only descriptive facts from the data table as input representations is sufficient to generate descriptive texts that explicitly mention facts in the table. However, since we intend to produce more analytical text with numerical reasoning, we propose adding inferred facts to the input representation by computing a set of arithmetic operations on the data table beforehand, defined as a pre-executed operation table.

Data Table We view T as a set of cells with their corresponding row header (rh), column header (ch), numerical value (val), and metric-type (m), defined as a data table (T_D). A data table for the example in Figure 1 consists of rh: ((model, our full model), (model, lee et al. (2018))); ch: (); val: ((89.6, 82.2, 85.7), (86.2, 83.7, 84.9); and m: (precision, recall, f1). Since our tables are annotated with a targeted header as a content plan for table descriptions, we mark cells corresponding to the targeted header with a target flag (tgt) to highlight the marked cells in text generation. We set $tgt = 1$ for targeted cells and $tgt = 0$ for non-targeted cells. In this study, we preprocess the header name by concatenating the row and column headers ($h = [rh; ch]$) and keep information about the header category by extracting overlapping tokens of row and column headers as th. As a result, we define $T_D = (h_{ij}, th_{ij}, val_{ij}, m_{ij}, tgt_{ij})$, where $1 \leq i \leq n_r$, $1 \leq j \leq n_c$; n_r and n_c are the numbers of rows and columns, respectively.

Pre-executed Operation Table We provide a table of pre-executed cell operations (T_{OP}) by doing mathematical operations only on targeted cells to limit the calculation. In this study, we cover maximum, minimum, and difference operations. Examples of a preprocessed table, data table, and pre-executed operation table are shown in Figure 2.

Linearized Table Supporting transfer learning of pre-trained transformers to our table-to-text generation task, we prepare a linearized table P_T as an input representation so that it similar to the representation that encoder has seen during pre-training. T is converted to a flat string $P_T = w_1, ..., w_{|P_T|}$, similar to that used in many prior work (Wang et al., 2020; Chen et al., 2020a; Kale and Rastogi, 2020b), where w_i denotes the i-th word in paragraph P_T with length $|P_T|$. In this study, we adopt the template-based input representation, introduced by Kale and Rastogi (2020a), to handle representation bias between a structured data T and a natural language utterance P_T, where P_T is generated using a manually defined template. We propose not only covering data table T_D in the template but also injecting the pre-executed numerical operations of table T through T_{OP} to guide numerical-reasoning-based text generation. We consider four different methods[3] for converting T into sequences, the last two being our contributions.

[3] An example is shown in Table 6 in the appendix.



		precision	recall	f1
model	our full model	89.6	82.2	85.7
model	lee et al.(2018)	86.2	83.7	84.9

header name		val	metric (m)	target
h	th			
our full model	model	89.6	precision	1
our full model	model	82.2	recall	1
our full model	model	85.7	f1	1
lee et al. (2018)	model	86.2	precision	0
lee et al. (2018)	model	83.7	recall	0
lee et al. (2018)	model	84.9	f1	0

op name	op arguments		metric	result	
	h	th		h	val
max	our full model, lee et al. (2018)	model	precision	our full model	89.6
max	our full model, lee et al. (2018)	model	f1	our full model	85.7
max	our full model, lee et al. (2018)	model	recall	lee et al. (2018)	83.7
min	our full model, lee et al. (2018)	model	recall	our full model	82.2
min	our full model, lee et al. (2018)	model	precision	lee et al. (2018)	86.2
min	our full model, lee et al. (2018)	model	f1	lee et al. (2018)	84.9
diff	our full model, lee et al. (2018)	model	precision		3.4
diff	our full model, lee et al. (2018)	model	recall		-1.5
diff	our full model, lee et al. (2018)	model	f1		0.8

Figure 2: Examples of preprocessed table, data table, and pre-executed operation table.

1. Naive Representation
   T is simply flattened into a sequence ignoring its table structure by concatenating captions, headers, metrics, and targeted cell values:

   caption: <table_id> <caption>. row name: <rh_1> ... <rh_{nr}>. column name: <ch_1> ... <ch_{nc}>. metric:<m_1>, ..., <$m_{nr/nc}$>. value: <$val_{1.1}$> ... <$val_{nr.nc}$> .

   This naive representation omits the relation between rows and columns. Note that <table_id> is extracted from the caption to support table mentioning in generating table descriptions.

2. Data-based Template (T_D temp)
   T is transformed into a natural language sentence by scanning each row of T_D with $tgt = 1$ to fill a manually defined template:

   <table_id> shows <caption>. <$m_{1.1}$> of <$h_{1.1}$> is <$val_{1.1}$> ... <$m_{nr.nc}$> of <$h_{nr.nc}$> is <$val_{nr.nc}$>.

   This representation covers the semantics of data in the original table.

3. Reasoning-based Template (T_{OP} temp)
   Mathematical operation arguments and results from T_{OP} are injected in this representation to cover the numerical reasoning of data in the original table. We define h_{op} and val_{op} as a header and a value of an operation result respectively, where op ={max, min, diff}. Specific to the difference operation, h_{diff1} and h_{diff2} refer to the first and second header arguments, respectively. Then, T is represented by concatenating the templatized representation for each row of T_{OP}:

   <table_id> shows <caption>. <h_{max}> has the largest <m_{max}> (<val_{max}>) of <th_{max}>. <h_{min}> has the smallest <m_{min}> (<val_{max}>) of <th_{min}>. <m_{diff}> of <h_{diff1}> is larger/smaller than <h_{diff2}> .

4. Data and Reasoning-based Template ($T_D + T_{OP}$ temp)
   T is converted by combining templatized sentences of T_D and T_{OP}. This representation covers both data and their numerical reasoning.

5 Generation Models

The task is to generate text by translating table representation P_T into table description $Y = y_1, y_2, ..., y_n$. We apply a series of generation models to solve the proposed task. While our focus is primarily on pre-trained models since they have been most widely used for limited data settings,

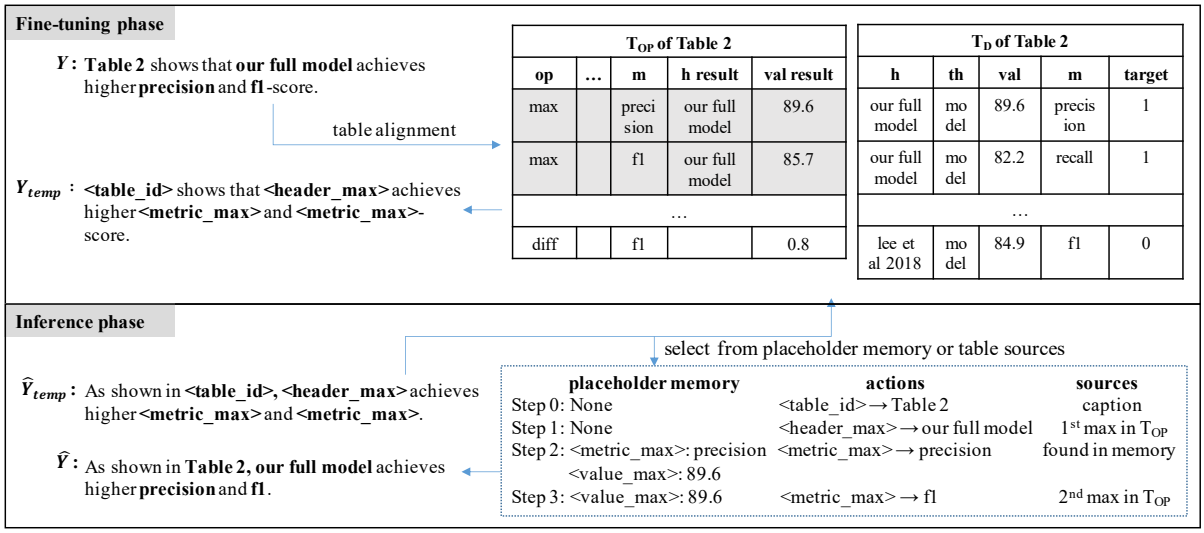


Figure 3: Placeholder alignment in copy-based pre-trained model.

like ours, we also include a template-based generator and a pointer-generator network as baselines.

5.1 Non-pre-trained Models

Template-based Generator We design a domain-specific template-based generator covering two types of sentences in producing table descriptions: table referring sentences and data description sentences. Since our task focuses on numerical-reasoning descriptions, we define templatized sentences using maximum records in table T_{OP}:

> <table_id> shows <caption>. we can see that <h$_{max}$> outperforms other <th$_{max}$> with <val$_{max}$> of <m$_{max}$>.

Pointer-Generator Pointer-generator (See et al., 2017) is a sequence-to-sequence model with attention and a copy mechanism. This model copes with the out-of-vocabulary problem in data-to-text generation by jointly copying from source texts and generating from a vocabulary.

5.2 Pre-trained Models

Fine-tuned GPT2 GPT2 (Radford et al., 2019) is a pre-trained language model with a decoder-only transformer architecture. We fine-tuned the GPT2 model by using table representation P_T as a prefix of our input. Specifically, we fed the concatenation of table representation P_T and table description Y to the model and generated Y. In the inference phase, we used only P_T as the input to generate $\hat{Y}$ starting after the last token of P_T.

Fine-tuned T5 T5 (Raffel et al., 2020) is a pre-trained transformer model with an encoder-decoder architecture, that solves natural language tasks by converting into a text-to-text format. We fine-tuned the T5 model in our dataset by adding a "summarize" prefix to table representation P_T producing output $\hat{Y}$.

Copy Mechanism Pre-trained language models have proven their effectiveness in handling the open vocabulary problem through subword tokenization. Supported by attention layers of the transformer in their architecture, the models learn to attend to source inputs while generating target texts in subword units. However, pre-trained generators often produce texts that are not aligned to table sources. In this study, we propose strengthening their copying ability by incorporating a copy mechanism into the pre-trained models. Although a copy mechanism based on pointer-generator (See et al., 2017) was used for pre-trained models (Chen et al., 2020c) and is well-known in the community, it cannot maintain the global logical structure of sentences with richer inference. We instead employed a simpler copy mechanism based on placeholders (Murakami et al., 2017) with more specific tags than in Chen et al. (2020a). We further propose a ranking-based placeholder alignment algorithm, as illustrated in Figure 3.

First, we align entities and numbers in Y with the data tables T_D and pre-executed arithmetic operation results T_{OP} by using string matching. The alignment starts from the first row to the last row of T_{OP}. If no matched token is found, it continues

Model	BLEU	ROUGE-L	METEOR	BERTSCORE	PARENT
Template-based	2.82	26.97	15.82	86.88	17.15
Pointer-generator (naive)	2.80	15.26	7.82	76.38	1.40
Fine-tuned GPT2 (naive)	3.06	23.7	18.84	85.12	6.56
Fine-tuned GPT2 (T_D temp)	3.01	22.97	*17.10	*84.68	6.53
Fine-tuned GPT2 (T_{OP} temp)	4.63	*25.39	18.85	*85.66	7.72
Fine-tuned GPT2 ($T_D + T_{OP}$ temp)	5.05	*25.13	19.14	*85.40	8.05
Fine-tuned GPT2 (naive) + Copy	1.29	*11.66	*6.94	*78.73	*2.45
Fine-tuned GPT2 (T_D temp) + Copy	1.36	*11.23	*6.43	*77.76	*2.10
Fine-tuned GPT2 (T_{OP} temp) + Copy	1.18	*9.40	*4.42	*73.83	*0.91
Fine-tuned GPT2 ($T_D + T_{OP}$ temp) + Copy	1.22	*9.62	*5.47	*70.87	*1.55
Fine-tuned T5 (naive)	4.25	29.71	18.94	87.64	13.09
Fine-tuned T5 (T_D temp)	5.02	30.25	*20.11	87.68	15.09
Fine-tuned T5 (T_{OP} temp)	4.99	28.63	18.85	*87.17	12.25
Fine-tuned T5 ($T_D + T_{OP}$ temp)	4.83	29.13	18.46	87.34	12.78
Fine-tuned T5 (naive) + Copy	5.14	*27.40	18.49	*86.37	*12.47
Fine-tuned T5 (T_D temp) + Copy	4.96	*27.08	18.23	*86.12	*11.65
Fine-tuned T5 (T_{OP} temp) + Copy	5.24	*28.02	18.68	*86.52	*11.96
Fine-tuned T5 ($T_D + T_{OP}$ temp) + Copy	5.45	*28.15	19.16	*86.54	*12.95

Table 2: Experimental results of different models with various types of table representations and proposed copy mechanism. Scores with asterisk * symbol were significantly different from those of naive models under Wilcoxon test ($p < 0.05$).

to the rows of T_D. We set a higher rank to T_{OP} than T_D in the alignment since we focus on logical text generation. Then, we replace the matched tokens with corresponding placeholders[4] in a templatized description Y_{temp}. As depicted in Figure 3, since "our full model" in sentence Y is matched with the header result of the maximum operation, we replace it with <header_max> placeholder. During the fine-tuning phase, instead of directly generating Y, the models learn to produce a templatized description Y_{temp} including placeholders as well as words.

In the inference phase, we design a ranking algorithm with a placeholder memory to select the best-replaced tokens for placeholders of a predicted templatized description $\hat{Y}_{temp}$ in producing a generated description $\hat{Y}$. We define a set of values in the same row of source tables as a content set and prioritize replacing placeholders in one sentence with the same content set, ensuring sentence coherence. A content set of T_D is a tuple of header, metric, and value. For T_{OP}, a content set consists of header, metric, and value of the operation results. Specific to the difference operation, we add the header of the first and second arguments to the content set since the header arguments are important to capture entity comparison in a sentence.

We utilize a placeholder memory to temporarily save prioritized placeholder candidates from the same content set that is previously chosen. For example, as shown in Figure 3, after replacing the header_max placeholder with the header result from the first row of maximum records of T_{OP} in Step 1, the related placeholders from the same content set (metric_max and value_max) are added to the placeholder memory as higher-ranked candidates in the searching space. The placeholder memory is reset to empty in the following sentence of $\hat{Y}_{temp}$ and the alignment starts again from the next content set of table sources.

6 Experiments

We conducted experiments on the proposed dataset to evaluate the performance of the text generation models and verify the effectiveness of the approach of using different table representations.

6.1 Automatic Evaluation Metrics

We used BLEU (Papineni et al., 2002), ROUGE-L (Lin, 2004), and METEOR (Banerjee and Lavie, 2005) to evaluate the informativeness of generated texts. We computed the BERTSCORE (Zhang et al., 2020) to assess the similarity between the generated texts and the ground-truth table descriptions by using contextualized token embeddings of pre-trained BERT (Devlin et al., 2019), which have been shown to be effective for paraphrase detection. Considering both references and table contents, we also used the PARENT metric, proposed by Dhingra et al. (2019). In our experiments, we modified the PARENT calculation by adding noun phrases of table captions as table contents and used only

[4]Details of placeholders and their definition are in Tables 7 and 8 in the appendix.

targeted table contents for table sources.

6.2 Implementation Details

We trained a pointer-generator model using the Adagrad optimizer with a batch size of 8 and a learning rate of 0.15. For fine-tuning the GPT2 model, the Adam optimizer set weight decay to 3×10^{-5}. Following Raffel et al. (2020), the T5 model was fine-tuned with a constant learning rate of 0.001. We trained all models for a maximum of ten epochs with early stopping based on the loss score on the validation set (patience of 3). At the time of decoding, the generated text was produced through a beam search of size 5.

7 Results

7.1 Automatic Evaluation

Table 2 shows our experimental results. The fine-tuned T5 models performed better than the others in terms of BLEU, ROUGE-L, METEOR, and BERTSCORE. The slightly lower PARENT of the best fine-tuned T5 model than the template-based generator implies that the fine-tuned T5 model was also comparable in terms of generating related table descriptions. The pointer-generator model had the lowest score since our dataset consists of limited table collections with a broad vocabulary and challenging target texts.

Effect of table representation Comparing the performance between table representation types in the pre-trained models, we can see a different tendency between GPT2 and T5. The more similar the table representation used as an input, the higher the score of GPT2. Since GPT2 had only a decoder, the inputs including reasoning-based templates (T_{OP} and $T_D + T_{OP}$), which are more similar to our target with numerical reasoning, performed the best for several metrics with more than 1 point improvement. In T5 with an encoder-decoder architecture, on the contrary, there was only a slight margin between different table representations. This indicates that the encoder part of T5 can capture table contexts from various input templates. For variants without a copy mechanism, T5 with only data representation (T_D) outperformed the other representation types with longer sentences for all metrics. Because of the gap between the encoder and decoder, T5 still had difficulty aligning the information of longer inputs and outputs.

Effect of copy mechanism The worst scores of the fine-tuned GPT2+copy models indicate that our proposed copy mechanism failed to learn the templatized target patterns in the fine-tuning step. The decoder-only GPT2 could not handle the sparse distributions of target texts with placeholders. Conversely, the copy-based fine-tuned T5 models achieved a better BLEU score due to their encoder and decoder ability in handling output texts with placeholders.

7.2 Qualitative Analysis

Table 3 shows table descriptions generated by the template-based, pointer-generator, and fine-tuned pre-trained models (GPT2 and T5), using data and reasoning-based templates[5] for our table example in Figure 2. We marked sentences related to table captions in green, correct facts based on table contents in blue, and incorrect facts in red. In this study, since we had a limited training set with a broader vocabulary, the pointer-generator model tended to result in repetitive words and failed to generate well-described descriptions. The pre-trained models, GPT2 and T5, generated more natural descriptions. While several pieces of text generated by GPT2 included numerical facts, they used numbers that were not extracted from table contents. The T5 models produced descriptions that were more related to table contents than GPT2.

Considering our lengthy output examples in Table 3, unlike the fine-tuned GPT2 model, which generated longer sentences, the fine-tuned T5 model generated shorter sentences than the references.[6] The length gap between the references and outputs of the fine-tuned T5 model affected the F1-based metrics of ROUGE-L, METEOR, BERTSCORE, and PARENT. Note that BLEU is a precision-based metric that can handle shorter outputs through a brevity penalty (Papineni et al., 2002). Therefore, we assume that BLEU better represents the performance of the fine-tuned T5 model than the other metrics.

7.3 Human Evaluation

We conducted a human evaluation[7] to better assess the quality of the generated text. We compared our copy-based fine-tuned T5 model with

[5] Examples using other table representations are shown in Table 9 in the appendix.

[6] Average token length of references: 80.57, GPT2: 87.39, GPT2+copy: 73.58, T5: 39.81, T5+copy: 41.81.

[7] The interfaces are shown in Figures 4 and 5 in the appendix.

Model	Generated Text
Reference	table 2 shows the mention detection results on the test set. similar to coreference linking results, our model achieves higher precision and f1 score, which indicates that our model can significantly reduce false positive mentions while it can still find a reasonable number of mentions.
Template-based	table 2 shows the overall mention detection results on the test set of ontonotes. we can see that our full outperforms other model with **85.7** f1.
Pointer-generator	the table 2 shows the results of the results of the results of the results of the the art of the art of...
Fine-tuned GPT2 ($T_D + T_{OP}$ temp)	the table shows the recall performance with our full model. the result of our full model is **88.7**, which is comparable with the **89.3** performance of our full model but still better than the **89.9** and **89.2** performance in both the f1, prec and full models. we also find that our full model does not perform very well when compared against a full one, with **89.4%** and **89.4%** recall and **89.2%** recall respectively. we can also find that our full model is slightly inferior in terms of recall.
Fine-tuned GPT2 ($T_D + T_{OP}$ temp) + Copy	table 2 : (the - d model : a) the : aa . the: the, the: and, the: the: the,the and, theand: the and, theand: theand: theand, theand: ... (<table_id>: table 2; <cat_header>: model)
Fine-tuned T5 ($T_D + T_{OP}$ temp)	table 2 presents the overall mention detection results on ontonotes. our full model outperforms all the state-of-the-art systems in terms of recall and f1 score.
Fine-tuned T5 ($T_D + T_{OP}$ temp) + Copy	table 2 shows the overall mention detection results on the test set of ontonotes. our full model outperforms the previous state-of-the-art models by a large margin, which confirms the effectiveness of our proposed approach. (<table_id>: table 2; <header_max>: our full model)

Table 3: Example of generated table description.

Model	Descriptive Facts			Inferred Facts			Relevance
	#Supp	#Cont	%Cont	#Supp	#Cont	%Cont	
Template-based	1.00	0.01	0.01	0.93	0.11	10.64	3.89
Pointer-generator	0.00	0.00	0.00	0.00	0.00	0.00	1.50
Fine-tuned GPT2	0.03	1.28	97.46	0.43	1.94	81.78	2.36
Fine-tuned T5	0.05	0.07	54.55	0.50	1.10	68.75	3.51
Fine-tuned T5 + Copy	0.04	0.04	50.00	0.78	0.57	42.62	3.78

Table 4: Average number of supporting and contradicting facts in generated table descriptions, percentage of contradicting to total facts, and levels of relevance to table captions.

Model	Gram	Coher	Conc
Template-based	7.78	11.11	−9.44
Pointer-generator	−72.78	−77.22	−78.33
Fine-tuned GPT2	31.11	28.89	27.78
Fine-tuned T5	18.33	17.22	39.44
Fine-tuned T5 + Copy	15.56	19.44	20.56

Table 5: Grammaticality, coherence, and conciseness levels of table description generators.

the template-based, pointer-generator, fine-tuned GPT2, and fine-tuned T5 models. We did not compare it against the copy-based fine-tuned GPT2 since GPT2 failed to incorporate our proposed copy mechanism. We used the best table representation with majority metrics for each model on the basis of the experimental results in Table 2.

In the first study, we evaluated the correctness of the generated text on the basis of facts in tables. We randomly selected 30 tables in the test set and elicited responses from three graduate students per table. Following Wiseman et al. (2017), the raters were asked to count how many facts in the descriptions were supported by numerical data in the tables and how many were contradicted. Since our task covers numerical-reasoning text, we distinguished descriptive numerical facts from inferred numerical facts. We also measured the level of relevance of the generated text to the table captions by using a four-point Likert scale (highly relevant, relevant, somewhat relevant, and irrelevant).

The results are shown in Table 4. The pointer-generator failed to reflect facts due to the wide variety of our table schemas. While the fine-tuned GPT2 model generated sentences with a larger number of descriptive and inferred facts than the others on average, most of the facts were contradictive. The fine-tuned T5 model generated fewer sentences than GPT2, with the average number of inferred facts being larger than that of descriptive facts. Our model based on the fine-tuned T5 model with a copy mechanism reduced the ratio of contradictive facts for both descriptive and inferred facts.

Following earlier work (Puduppully et al., 2019),

we also evaluated text fluency in terms of grammaticality, coherence, and conciseness by using best-worst scaling (BWS) (Louviere and Woodworth, 1991; Louviere et al., 2015). We divided the outputs of the five models into ten pairs of descriptions. We presented workers with two descriptions and asked them to decide which one is best for each fluency category.

The score of each model was calculated by using the MaxDiff approach (Orme, 2009): the number of times a description was chosen as the best minus the number of times it was chosen as the worst. Scores range from -100 (absolutely worst) to 100 (absolutely best). We elicited judgments with Amazon Mechanical Turk for the 30 descriptions, rated by 3 participants. The results are shown in Table 5. Most of the pre-trained models achieved better scores than the others. The fine-tuned GPT2 model achieved the highest score in terms of grammaticality and coherence. The fine-tuned T5 model achieved the highest score in terms of conciseness. Adding a copy mechanism to the T5 slightly decreased the grammaticality and conciseness but improved the coherence.

8 Conclusion

We proposed numericNLG, a new dataset for table-to-text generation using a table and its corresponding description from scientific papers, focusing on numerical-reasoning texts. Even though our proposed dataset is not a large-scale table collection, we provided pairs of a table and its rich inference description, that are naturally written by experts in scientific papers, supporting further research on table-to-text generation with numerical reasoning.

We conducted experiments with fine-tuned pre-trained models by using several types of table linearization as input representations, comparing with a template-based generator and pointer-generator. The experiments showed that transfer-learning of pre-trained language models leads to an improvement in our settings, that resulted in more fluent text while it still lacked fidelity to table contents. We then proposed incorporating a copy mechanism by using general placeholders to avoid the production of hallucinated phrases, that are not supported by tables while preserving high fluency. Even though our proposed copy mechanism failed to learn to generate better outputs in the decoder-only pre-trained models, we showed that a copy-based pre-trained model with an encoder-decoder architecture leads to a better BLEU score and improves correctness.

Acknowledgements

Lya Hulliyyatus Suadaa is supported by the Indonesian Endowment Fund for Education (LPDP) and the Okumura-Takamura-Funakoshi Laboratory, Tokyo Institute of Technology. This work is partially supported by JST PRESTO (Grant Number JPMJPR1655). We thank the anonymous reviewers for their helpful discussion on this work and comments on the previous draft of the paper.

References

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Wenhu Chen, Jianshu Chen, Yu Su, Zhiyu Chen, and William Yang Wang. 2020a. Logical natural language generation from open-domain tables. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7929–7942, Online. Association for Computational Linguistics.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020b. Tabfact : A large-scale dataset for table-based fact verification. In *International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia.

Zhiyu Chen, Harini Eavani, Wenhu Chen, Yinyin Liu, and William Yang Wang. 2020c. Few-shot NLG with pre-trained language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 183–190, Online. Association for Computational Linguistics.

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.

Bhuwan Dhingra, Manaal Faruqui, Ankur Parikh, Ming-Wei Chang, Dipanjan Das, and William Cohen. 2019. Handling divergent reference texts when evaluating table-to-text generation. In *Proceedings of the 57th Annual Meeting of the Association for*

Computational Linguistics, pages 4884–4895, Florence, Italy. Association for Computational Linguistics.

Vivek Gupta, Maitrey Mehta, Pegah Nokhiz, and Vivek Srikumar. 2020. INFOTABS: Inference on tables as semi-structured data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2309–2324, Online. Association for Computational Linguistics.

Mihir Kale and Abhinav Rastogi. 2020a. Template guided text generation for task-oriented dialogue. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6505–6520, Online. Association for Computational Linguistics.

Mihir Kale and Abhinav Rastogi. 2020b. 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.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213, Austin, Texas. Association for Computational Linguistics.

Percy Liang, Michael Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 91–99, Suntec, Singapore. 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.

Jordan J. Louviere, Terry N. Flynn, and A. A. J. Marley. 2015. Best-worst scaling: A model for the largest difference judgments. In *Cambridge University Press*.

Jordan J. Louviere and George G. Woodworth. 1991. Best-worst scaling: A model for the largest difference judgments. In *University of Alberta: Working Paper*.

Soichiro Murakami, Akihiko Watanabe, Akira Miyazawa, Keiichi Goshima, Toshihiko Yanase, Hiroya Takamura, and Yusuke Miyao. 2017. Learning to generate market comments from stock prices. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1374–1384, Vancouver, Canada. Association for Computational Linguistics.

Feng Nie, Jinpeng Wang, Jin-Ge Yao, Rong Pan, and Chin-Yew Lin. 2018. Operation-guided neural networks for high fidelity data-to-text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3879–3889, Brussels, Belgium. Association for Computational Linguistics.

Bryan Orme. 2009. Maxdiff analysis: Simple counting, individual-level logit, and hb. In *Sawtooth Software, Inc.*

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. 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.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-text generation with entity modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2023–2035, Florence, Italy. Association for Computational Linguistics.

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

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

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.

Lya Hulliyyatus Suadaa, Hidetaka Kamigaito, Manabu Okumura, and Hiroya Takamura. 2021. Metric-type

identification for multi-level header numerical tables in scientific papers. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3062–3071, Online. Association for Computational Linguistics.

Hao Wang, Xiaodong Zhang, Shuming Ma, Xu Sun, Houfeng Wang, and Mengxiang Wang. 2018. A neural question answering model based on semi-structured tables. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1941–1951, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Zhenyi Wang, Xiaoyang Wang, Bang An, Dong Yu, and Changyou Chen. 2020. Towards faithful neural table-to-text generation with content-matching constraints. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1072–1086, Online. Association for Computational Linguistics.

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.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

A Table Representation

An example of table representation for Figure 2 is shown in Table 6.

Type	Table Representation
Naive	caption: table 2 the overall mention detection results on the test set of ontonotes. row name: model our full, model lee et al. (2018). metric: prec., rec., f1. value: 89.6 82.2 85.7.
T_D temp	table 2 shows the overall mention detection results on the test set of ontonotes . prec. of model our full model is 89.6 . rec. of model our full model is 82.2 . f1 of model our full model is 85.7 .
T_{OP} temp	table 2 shows the overall mention detection results on the test set of ontonotes . our full model has the largest prec. (89.6) of model. lee et al. (2018) has the largest rec. (83.7) of model . our full model has the largest f1 (85.7) of model . prec. of model our full model is larger than model lee et al. (2018) . rec. of model our full model is smaller than model lee et al. (2018) . f1 of model our full model is larger than model lee et al. (2018) .
T_D+ T_{OP} temp	table 2 shows the overall mention detection results on the test set of ontonotes . prec. of model our full model is 89.6 . rec. of model our full model is 82.2 . f1 of model our full model is 85.7 . model our full model has the largest prec. (89.6) . model lee et al. (2018) has the largest rec. (83.7) . model our full model has the largest f1 (85.7) . prec. of model our full model is larger than model lee et al. (2018) . rec. of model our full model is smaller than model lee et al. (2018) . f1 of model our full model is larger than model lee et al. (2018) .

Table 6: Example of table representation.

B Placeholders of Copy-based Pre-trained Models

Tables 7 and 8 describe placeholders of our proposed copy-based pre-trained models.

Placeholder	Description	Example of Figure 2
<header_target>	targeted header	our full model
<metric_target>	metric of targeted header	precision
<value_target>	value of targeted header	89.6
<header_other>	non-targeted header	lee et al. (2018)
<metric_other>	metric of non-targeted header	precision
<value_other>	value of non-targeted header	86.2
<cat_header>	header category	model
<metric>	metric lists	precision, recall, f1

Table 7: General placeholder descriptions for T_D.

Placeholder	Description	Example of Figure 2
<header_max>	header of maximum records	our full model
<cat_header_max>	header category of maximum records	model
<metric_max>	metric of maximum records	precision
<value_max>	maximum value of records	89.6
<header_min>	header of minimum records	our full model
<cat_header_min>	header category of minimum records	model
<metric_min>	metric of minimum records	recall
<value_min>	minimum value of records	82.2
<header_diff1>	the first header argument of difference records	our full model
<header_diff2>	the second header argument of difference records	lee et al. (2018)
<cat_header_diff>	header category of difference records	model
<metric_diff>	metric of difference records	precision
<value_diff>	difference value of records	3.4

Table 8: General placeholder descriptions for T_{OP}.

C System Output Examples

Table 9 shows table descriptions generated by the fine-tuned GPT2 and fine-tuned T5 models with and without a copy mechanism, using different types of table representations for our table example in Figure 2.

D Human Evaluation

Figures 4 and 5 show the user interface for evaluating correctness and relevance and for evaluating grammaticality, coherence, and conciseness, respectively.

Model	Generated Text
Fine-tuned GPT2 (naive)	we compare our results using the model with that of the other model, which shows the performance on both models. we can conclude that the models are more reliable in predicting the usefulness of the model and more reliable than the other models. we suggest that the more accurate the model, the higher the mention detection results on both models. the model with the highest mentions detection yields a better model, and also shows the best performance on both models.
Fine-tuned GPT2 (T_D temp)	and finally we have seen that our model does not do the right thing by simply using the word "we." as a noun we do not do the right thing by using the verb "we." as a noun our word "we" does not even have a verb, "we do." as a verb we do not even have a verb, "we do.
Fine-tuned GPT2 (T_{OP} temp)	the table shows the recall results for our full model. the f1 performance was better than that in our full model, indicating that we did not have any other models in the dataset. the rec. performance was comparable to that for the full model with a f1+ performance of **82.7%**.
Fine-tuned GPT2 (naive) + Copy	table 2 a new version of the first game of a new series , f1 a new version of a game of an a new series , the new and another version of the original character , the one of the character of the character of the person of the person of the character of a person , ...
Fine-tuned GPT2 (T_D temp) + Copy	table 2 the best way to lee et al. (2018) the best for the . the first time, in the history, a. the, it has. is that the , the , the 86.2 . the, the.
Fine-tuned GPT2 (T_{OP} temp) + Copy	, model in our full and 3.4 . model . (,and, (and in) and inand(. (i. in)
Fine-tuned T5 (naive)	table 2 presents the overall mention detection results on the test set of ontonotes. we can see that our full model outperforms all the baselines in terms of recall f1 score.
Fine-tuned T5 (T_D temp)	table 2 shows the mention detection performance on the test set of ontonotes. our full model outperforms all the baselines in terms of recall and f1 score.
Fine-tuned T5 (T_{OP} temp)	table 2 shows the overall mention detection results on the test set of ontonotes. our full model outperforms the state-of-the-art in both precision and recall.
Fine-tuned T5 (naive) + Copy	table 2 shows the overall model results on ontonotes. we can see that our full model outperforms all baselines, which demonstrates the effectiveness of our approach.
Fine-tuned T5 (T_D temp) + Copy	table 2 shows the overall mention detection results on the test set of ontonotes dataset. our full model outperforms the state - of - the - art by a large margin , with an absolute difference of **0.8%** over the state of the art.
Fine-tuned T5 (T_{OP} temp) + Copy	table 2 shows the overall mention detection results on the test set of ontonotes. our model outperforms the state - of - the - art (lee et al . , 2018) and is comparable to the state - of - the - art (lee et al . , 2018).

Table 9: Example of generated table description.

Instructions (click to expand)

Article Title : Improving the Transformer Translation Model with Document-Level Context

Table 4: Comparison with Transformer on FrenchEnglish translation task.

	Dev	Test
Method || Transformer	29.42	35.15
Method || this work	30.40	36.04

Metric: BLEU
the higher the score, the better the performance.

Description	Descriptive Numerical Facts		Inferred Numerical Facts		Relevance Level
	Supportive	Contradictive	Supportive	Contradictive	
table 4 shows the results on french english translation task . we find that transformer outperforms all other methods significantly.	0	0	0	0	○ highly relevant ○ relevant ○ somewhat relevant ○ not relevant ○ not available
table 4 shows the comparison with transformer on frenchenglish translation task. we can see that our proposed model outperforms transformer by a large margin and achieves significant improvements over the baseline.	0	0	0	0	○ highly relevant ○ relevant ○ somewhat relevant ○ not relevant ○ not available

Table Related Question
What is the highest score for this work in Dev set?
○ 30.40 ○ 35.15 ○ 36.04 ○ 36.40

Submit

Figure 4: Interface for evaluating correctness and relevance.

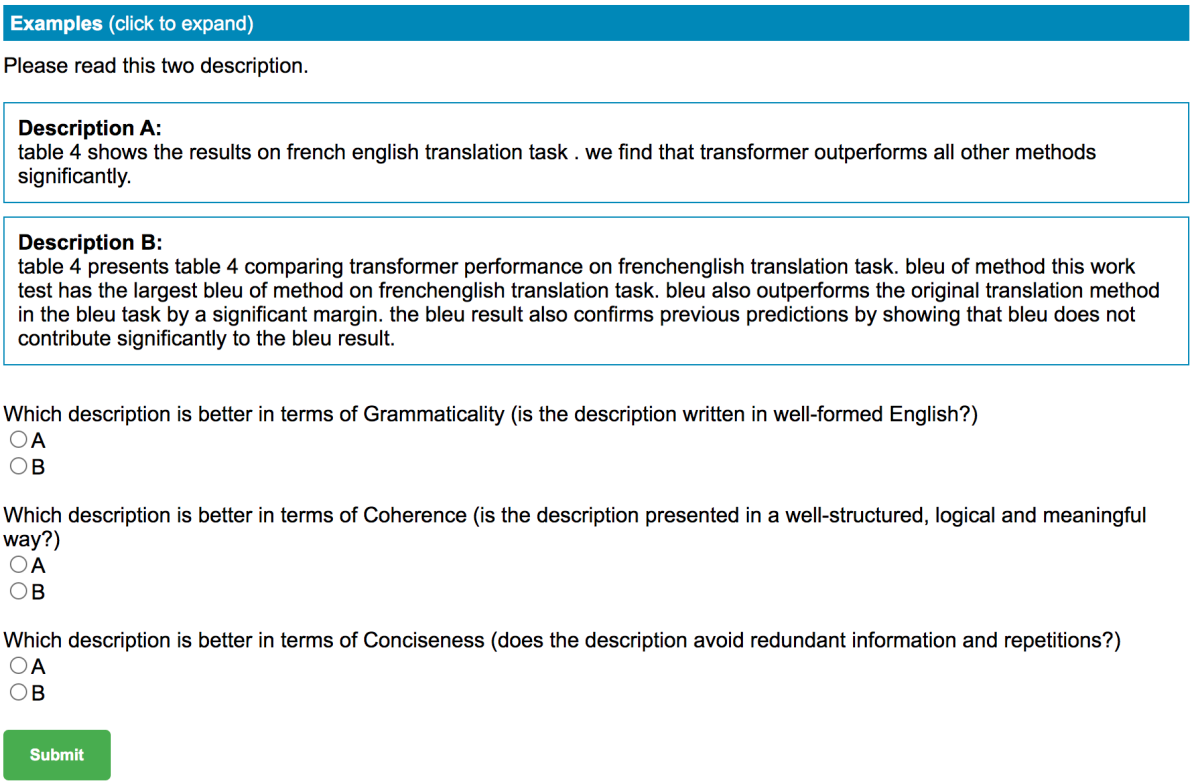


Figure 5: Interface for evaluating grammaticality, coherence, and conciseness.