FLAP: Table-to-Text Generation with Feature Indication and Numerical Reasoning Pretraining

Anonymous ACL submission

Abstract

Recent neural models have shown success in table-to-text generation. However, the performance of content selection and content planning is still unsatisfactory. In this paper, we propose an effective framework with **F**eature indication and numerica**L** re**A**soning **P**retraining (**FLAP**) to help the neural generation model on content selection and planning. FLAP is an end-to-end generation model that takes the whole table as input and utilize explicit content selection indication with the feature indication mechanism to ensure consistency between training and inference. As numerical reasoning plays a crucial role in both content selection and planning. Rather than treating the table as a sequence of token embeddings, we treat values of a table as scalars and map the whole table into a numerical vector for explicit content selection with machine learning algorithms. Additionally, we design a QA-based numerical reasoning pretraining task to enhance numerical reasoning ability of our pretrained model. Experiments show that our framework outperforms the strong baselines on metrics of both content selection and planning on ROTOWIRE and RW-FG. Thorough analyses demonstrate the effectiveness of our proposed method.

PLAYER_NAME	PTS	AST	REB	STL	BLK	TEAM_CITY
Rashad Vaughn	0	1	0	0	0	Milwaukee
Steve Novak	4	0	0	0	0	Milwaukee
Tyler Ennis	3	0	0	0	0	Milwaukee
Johnny O'Bryant III	0	0	1	0	0	Milwaukee
Giannis Antetokounmpo	8	7	12	0	0	Milwaukee
Miles Plumlee	**10**	0	**6**	**1**	**2**	Milwaukee
Jabari Parker	**13**	**3**	**5**	0	0	Milwaukee
O.J. Mayo	9	2	2	0	0	Milwaukee
Khris Middleton	**26**	**2**	**7**	**1**	**1**	Milwaukee
Greg Monroe	**10**	**3**	**7**	**1**	**1**	Milwaukee
Michael Carter-Williams	8	3	4	0	3	Milwaukee
Marcus Morris	**20**	**2**	**8**	0	0	Detroit
Tobias Harris	**15**	**4**	**4**	**1**	0	Detroit
Kentavious Caldwell-Pope	12	2	3	2	0	Detroit
Andre Drummond	**15**	**2**	**17**	**4**	2	Detroit
Reggie Bullock	8	0	4	2	0	Detroit
Reggie Jackson	**22**	**8**	**2**	**2**	0	Detroit
Aron Baynes	2	1	5	1	0	Detroit
Steve Blake	0	4	2	1	0	Detroit
Justin Harper	0	0	0	0	0	Detroit
Darrun Hilliard	8	1	1	0	0	Detroit

…**Jackson** paced the Pistons' attack with 22 points, eight assists, two rebounds and two steals. **Marcus Morris** followed with 20 points, eight rebounds and two assists. **Tobias Harris** and **Drummond** were next with a pair of 15-point efforts. The former added four rebounds, four assists and a steal, while the latter hauled in 17 boards, dished out two assists, recorded four steals and registered a pair of blocks... Milwaukee was led by **Khris Middleton**'s 26 points, which he supplemented with seven rebounds, two assists, a steal and a block. **Jabari Parker** followed with 13 points, five rebounds and three assists. **Miles Plumlee** continued to start at center over Greg Monroe, and collected 10 points, six rebounds, a steal and two blocks. **Monroe** paced the reserves with 10 points, seven rebounds, three assists, a steal and a block…

PTS: Points AST: Assist REB: Rebound STL: Steal BLK: Block

Figure 1: An example of a pair of game statistical table and its corresponding summary.

1 Introduction

Table-to-text generation is the task of taking data of table as input and producing proper and fluent text as output. An example can be seen in Figure 1. There are two basic procedures to perform table-to-text generation: content selection, which aims to select an appropriate subset of the input table, and then, content planning, which aims to transform the selected content into fluent natural text (Reiter and Dale, 1997; Gatt and Krahmer, 2018).

One line of works approaches this task in a pipeline fashion which first employs a model to select a certain part of the table and then maps the selected table content to its translated natural language with a seq2seq model (Puduppully et al., 2019a; Puduppully and Lapata, 2021; Gong et al., 2020). The apparent drawback of the pipeline approach is at inference stage, content planning is subject to the quality of the predicted upstream selected content, which results in exposure bias (Ranzato et al., 2016). Another line of works approaches the task in an end-to-end fashion which utilizes the seq2seq model to take the whole table as input and generates the summary (Puduppully et al., 2019b; Rebuffel et al., 2020; Gong et al., 2019; Li et al., 2021). These methods avoid the exposure bias problem by considering the whole table. However, it is very challenging to learn implicit content selection for the model and the absence of explicit selection indication might hinder the overall performance. Given the above, we identify end-to-end modeling with whole table input and explicit selection indication as the pivotal premised approach for our work and introduce a feature indication mechanism that treats predictions of the content selector as auxiliary features then feed the whole table as well as the auxiliary features into the seq2seq model.

We further make a key observation that numerical reasoning is crucial for both content selection and content planning. As highlighted in Figure 1, players with competitive performance, i.e. higher points or assists, tend to get mentioned in the summary. The value of the numbers in the statistical table greatly determines whether and how the records should be presented. Therefore, we propose an effective framework with numerical feature selection indication and numerical reasoning pretraining to enhance the numerical reasoning ability and improve the performance of content selection and planning, respectively. Specifically, for content selection, most previous methods treat the numerical values in the table as sequence tokens in the form of distributed representations which makes it inefficient to distinguish their relative magnitude. We propose to treat the numerical values as scalars directly and map the whole table into a numerical vector where each dimension corresponds to a certain record. This allows the traditional machine learning tools such as XGBoost (Chen and Guestrin, 2016) or Random Forest (Ho, 1995) that deal with numerical features to perform multi-label classification. The resulted predicted record features can be served as the content selection indication for the generation model. As for content planning, we introduce the pretraining and finetuning paradigm and design a series of numerical reasoning pretraining tasks that require the model to perform absolute or relative comparison and sorting of the numerical records in the table. Concretely, we construct the QA-based pretraining task to incorporates numerical reasoning ability to the seq2seq generation model and finetune the model on the downstream table-to-text task.

We conduct experiments on two document level table-to-text generation datasets ROTOWIRE (Wiseman et al., 2017) and RW-FG (Wang, 2019a). Our method achieves favorable improvements over several strong baselines in content selection precision, F1, content ordering (CO) and BLEU metrics. Experiments results demonstrate the effectiveness of our feature indication mechanism in dealing with exposure bias and the effectiveness of our numerical reasoning pretraining paradigm. Focusing on the numerical feature of the table and may also inspire future research.

2 Related Work

Table-to-text generation is the task of generating fluent text that properly describes the input table. The main challenge lies in content selection and planning (Reiter and Dale, 1997; Gatt and Krahmer, 2018). Recently, several neural generation systems have been proposed for table-to-text generation. One line of works follow a pipeline paradigm, Perez-Beltrachini and Lapata (2018) equip the sequence-to-sequence model with a content selection component that selects key records from the table then map the selected records into natural language. Puduppully et al. (2019a) additionally introduce a content planning module that sort the selected records. Gong et al. (2020) introduce a rank pretraining task to refine the value representation and optimize content selection and planning via policy gradient (Sutton et al., 1998). Puduppully and Lapata (2021) groups the selected records into a set of paragraph plans, then employ a content planning module to sort the paragraph plans. These methods take a subset of the table as input and the content selection at inference stage inevitably introduces error, which causes exposure bias. Other works perform table-to-text generation in a end-to-end fashion that take full table of records as input and directly generate the target sequence. Gong et al. (2019) propose a hierarchical encoder with dual attention to consider both the table structure and history information. Puduppully et al. (2019b) create dynamically updated entity-specific representations according to entity-centric theories (Grosz et al., 1994; Mann and Thompson, 1988). Rebuffel et al. (2020) propose a hierarchical transformer encoder to encode the table at both element level and structure level. Li et al. (2021) introduce a reasoning module and two supervision task on the encoder to capture the relations among records. These end-to-end models avoid the exposure bias problem but choose to model the content selection implicitly. Among them, Gong et al. (2019); Puduppully et al. (2019b); Li et al. (2021) utilize conditional copy mechanism to apply supervision to the switch gate, which make use of the content selection supervision during training.

Our work performs end-to-end modeling and takes explicit feature indication for content selection during both training and inference. Compared with previous works, we treat values of a table as scalars and map the whole table into a numerical vector. So we can model the content selection as a multi-label classification task, which is suitable for traditional ML algorithms such as XGBoost. The proposed numerical reasoning pretraining task and the use of pretrained model BART also encourage future work to consider large pretrained LMs.

3 Method

We start the section with the formulation of the table-to-text task. Then, we show how to construct numerical features and labels given a table to train our content selector for feature indication. Then we introduce our neural generation model with feature indication mechanism. Finally, we describe the numerical reasoning pretraining process. The input of our model is a table of records as shown in Figure 1. Each record is either a numerical value (e.g. player's points) or categorical value (e.g. player's starting position). Each row of the tables describes the statistics of an entity (a player or one of the two teams). The generation model needs to generate a summary that properly describes the table content. The output summary is a sequence of tokens $\mathrm{y} = \mathrm{y}_1\mathrm{y}_2...\mathrm{y}_{|\mathrm{y}|}$.

3.1 Feature & Label Construction

Previous works (Perez-Beltrachini and Lapata, 2018; Puduppully et al., 2019a; Puduppully and Lapata, 2021) treat the values in the table as sequence tokens and learn a distributed representations for content selection, which is inefficient and loses the numbers' exact value information. The numerical information is of great importance when it comes to choosing which record to mention. Rooted in this intuition, we treat the record values in the table as a vector $\mathbf{X}$ with scalar feature at each dimension. The corresponded label is $\mathbf{Label} = [1, 1, 0, 0, ...]$, in which each dimension indicates whether the corresponded record in the table is mentioned in the summary. This allows us to model content selection as a multi-label classification task.

As shown in Figure 3, we concatenate the scalar values in the table into a vector. Since each record can refer to a unique entity (a player or a team), we exclude the name of the player, team and the city. Other categorical values are also mapped into scalars. For example, we map START_POSITION from $\{\mathrm{P}, \mathrm{F}, \mathrm{G}\}$ into $\{0, 1, 2\}$. We use 0 to indicate a home player and 1 to indicate a visitor player. To construct $\mathbf{Label}$, we use an public available pretrained extracted model to extract the ground-truth records from the corresponding summary. Then we aligned the extracted records with records in the table and map the aligned records to $\mathbf{1}$ and others to $\mathbf{0}$. We train a traditional machine learning model XGBoost with multi-label classification objective as our content selector.

3.2 Generation Model

Our overall generation model follows a transformer (Vaswani et al., 2017) encoder-decoder architecture. The input of the encoder are three sequences of tokens: {K,V,F}. K=[ENT]k_{11},...,[ENT]k_{21}..., where k_{ij} represents the attribute of i-th row and j-th column of the table. V=[ENT]v_{11},...,[ENT]v_{21}..., where v_{ij} represents the value of i-th row and j-th column of the table. F=[ENT]f_{11},...,[ENT]f_{21}..., where $\mathrm{f}_{ij} \in \{\mathrm{F}, \mathrm{T}\}$ represents a feature token that indicates whether record of i-th row and j-th column of the table is mentioned in the summary. [ENT] is a special token that separates each records. In the training stage, we use the ground truth F, while in the inference stage, we use F predicted by the XGBoost content selector.

3.2.1 Encoding with Feature Indication

We first feed three sequences into a word embedding layer and get three embedding matrices:

$$\mathbf{E}_K, \mathbf{E}_V, \mathbf{E}_F = \mathrm{Emb}(\mathrm{K}), \mathrm{Emb}(\mathrm{V}), \mathrm{Emb}(\mathrm{F}), \tag{1}$$

where $\mathrm{Emb}(\cdot)$ is the word embedding layer with parameter $\mathbf{W} \in \mathbb{R}^{|V|*D}$, $|V|$ is the vocabulary size, D is the hidden size. The final representation of the embedding layer is:

$$\mathbf{H}_0 = (\mathbf{E}_K + \mathbf{E}_V + \mathbf{E}_F)/3 + \mathbf{P}, \tag{2}$$

where $\mathbf{P}$ is the position embedding matrix. Thus the feature indication F are fused into the representation $\mathbf{H}_0$ and the generation model is supervised with explicit content selection indication. Then $\mathbf{H}_0$ is fed into a transformer encoder:

$$\mathbf{H}_L = \mathrm{TransEncoder}(\mathbf{H}_0), \tag{3}$$

where $\mathbf{H}_L$ is the contextual representation, L is the number of encoder layers.

Notice that we input the whole table into the encoder along with feature indication sequence F

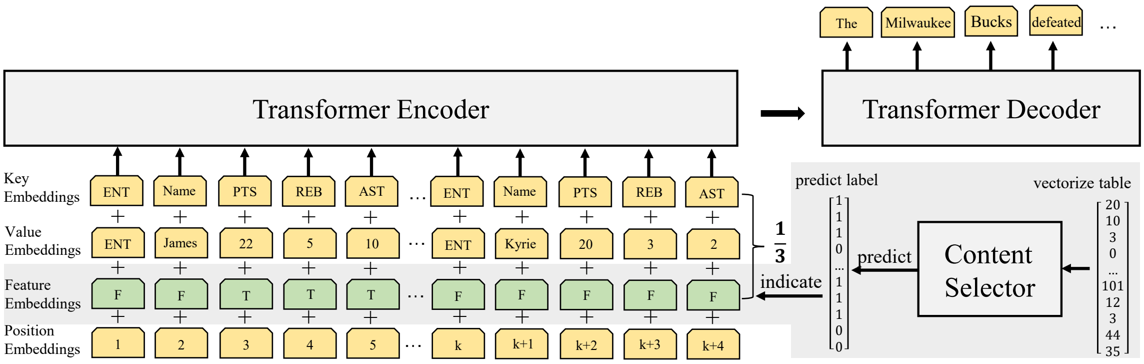

Figure 2: Overall model architecture.

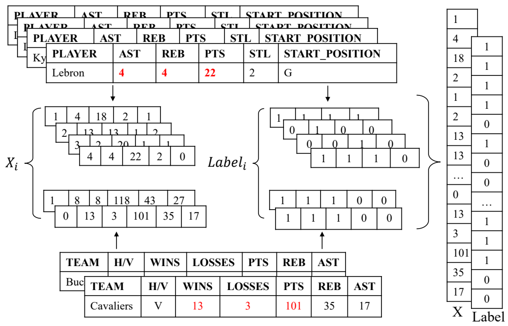

Figure 3: The feature construction process from a table. Values that occur in the summary are highlighted in red.

rather than only the selected subset of the table such that the model is not reduced to a translation model. Another motivation behind this is to fix the combination of table embeddings and prevent model from fitting the combination distribution to mitigate exposure bias problem in the inference stage. Specifically, in the inference stage, $\mathbf{E}_K, \mathbf{E}_V, \mathbf{P}$ as well as the shape of the initial representation $\mathbf{H}_0$ will not change, the only variation is reduced into the sub component $\mathbf{E}_F$.

3.2.2 Decoding and Objective Function

The output of encoder are then fed into the transformer decoder to predict next words one-by-one:

$$p(y_i|y_{1:i-1}, \mathbf{H}_L) = \text{TransDecoder}(\mathbf{H}_{y_{1:i-1}}, \mathbf{H}_L), \tag{4}$$

$$\mathbf{H}_{\mathbf{y_{1:i-1}}} = \text{Emb}(y_{1:i-1}) + \mathbf{P}_{y_{1:i-1}}, \tag{5}$$

where $\mathbf{P}_{\mathbf{y_{1:i-1}}}$ is the postition embedding of previous $i-1$ words. The final objective function is:

$$\mathbb{L}(\theta) = -\sum_{i=1}^{L} \log p(y_i|y_{1:i-1}, \mathbf{H}_L). \tag{6}$$

3.3 Numerical Reasoning Pretraining

3.3.1 Query Construction

As discussed above, numerical reasoning is also important for content planning. Inspired by the achievements of transfer learning (Torrey and Shavlik, 2010; Devlin et al., 2019), we propose to pretrain our model on a numerical reasoning pretraining task, then finetune on the downstream table-to-text generation task. According to the philosophy of transfer learning, the upstream task needs to share common knowledge with the downstream task. To this end, we design a series of QA-based numerical reasoning tasks that require the model to make absolute or relative comparison and sorting of the numerical records in the table. As detailed in Figure 4, for each table, we have six types of questions. For each question, we sample its corresponding arguments from the table. For example, for the question SORT, we need to sample n names and an attribute, and sort the names with respect to the attribute. In our experiments, we construct 150 questions and answers for each table in the training dataset. We obtain around 500,000 (table, query, answer) triples in total for our pretraining dataset. Each question and its corresponding answer are tokenized into a sequence.

3.3.2 Pretraining

For numerical reasoning pretraining, we use the same transformer encoder-decoder architecture. The model answers questions in a sequence-to-sequence manner. The input of the encoder are two sequences, one is the concatenation of table and the query attribute sequences: K_q=[ENT]k_{11},...,[ENT]k_{21}...[ENT]k_{q_1} ..., where k_{q_i} is the attribute token of the query sequence. The other is the concatenation of value sequences: V_q=[ENT]v_{11},...,[ENT]v_{21}...[ENT]v_{q_1} ..., where

TEAM	WINS	LOSSES	PTS	REB	AST	...
Bucks	8	8	118	43	27	...
Cavaliers	13	3	101	35	17	...

PLAYER	PTS	AST	REB	STL	CITY	...
Giannis	34	5	12	5	Milwaukee	...
Jabari	18	1	4	2	Milwaukee	...
Michael	17	2	2	1	Milwaukee	...
Lebron	20	4	4	0	Cleveland	...
Kyrie	10	3	2	1	Cleveland	...
Kevin	13	2	13	1	Cleveland	...

Query:	MAX(Giannis, Kyrie, Kevin, PTS)
Ans:	Giannis
Query:	MAX_NUM(Giannis, Kyrie, Kevin, PTS)
Ans:	34
Query:	EXCEED(Giannis, Kevin, Lebron, PTS, 20)
Ans:	Giannis, Lebron
Query:	EXCEED_NUM(Giannis, Kevin, Lebron, PTS, 20)
Ans:	34,22
Query:	SORT(Giannis, Jabari, Kevin, PTS)
Ans:	Kevin, Jabari, Giannis
Query:	SORT_NUM(Giannis, Jabari, Kevin, PTS)
Ans:	13,18,34

1. **MAX(Name1,Name2,...,K)** : Output player's name with the maximum value of attribute K.
2. **MAX_NUM(Name1,Name2,...,K)** : Output the maximum value of the attribute K.
3. **EXCEED(Name1,Name2, ..., K, V)** : Output players' names with attributes K higher than V.
4. **EXCEED_NUM(Name1,Name2, ..., K, V)** : Output the values that higher than V.
5. **SORT(Name1, Name2, ..., K)** : Sort the names of players according to values of attribute K.
6. **SORT_NUM(Name1, Name2, ..., K)** : Sort the values of players' attribute K.

Figure 4: Illustration of query and answer construction condition on the table of a game.

v_{q_i} is the i-th token in the query sequence. The output of the decoder is the corresponding sequential answer $\mathrm{y}^{\mathrm{q}} = \mathrm{y}_1\mathrm{y}_2...\mathrm{y}_{|\mathrm{y}^{\mathrm{q}}|}$. Similarly, we first feed the two sequence into a embedding layer:

$$\mathbf{E}_{K_q}, \mathbf{E}_{V_q} = \mathrm{Emb}(\mathbf{K}_q), \mathrm{Emb}(\mathbf{V}_q), \tag{7}$$

the initial representation is:

$$\mathbf{H}_0^q = (\mathbf{E}_{K_q} + \mathbf{E}_{V_q})/2 + \mathbf{P}_q, \tag{8}$$

where $\mathbf{P_q}$ is the position embedding matrix. Then, we feed $\mathbf{H}_0^q$ into the transformer encoder to get the contextual representation $\mathbf{H}_L^q$.

As for decoding, the answer sequence is decoded one-by-one:

$$p(y_i^q | y_{1:i-1}^q, \mathbf{H}_L) = \mathrm{TransDecoder}(\mathbf{H}_{y_{1:i-1}}^q, \mathbf{H}_L^q), \tag{9}$$

$$\mathbf{H}_{y_{1:i-1}}^q = \mathrm{Emb}(y_{1:i-1}^q) + \mathbf{P}_{y_{1:i-1}}^q, \tag{10}$$

where $\mathbf{P^q_{y_{1:i-1}}}$ is the postition embedding matrix of previous $i-1$ words. The objective function is:

$$\mathbb{L}^q(\theta) = -\sum_{i=1}^{L^q} \log p(y_i^q | y_{1:i-1}, \mathbf{H}_L^q). \tag{11}$$

4 Experiment

4.1 Dataset and Automatic Evaluation

We conduct our experiments mainly on the NBA game table-to-text datasets ROTOWIRE (Wiseman et al., 2017). The number of the attribute in the table is 39 and average length of summaries is 337. We also conduct experiments on RW-FG (Wang, 2019b), which is an enriched and cleaned version of ROTOWIRE which removes ungrounded sentences in the summary and add additional information into the table, such as the arena name. The dataset splits are 3,398/727/728 and 5,232/1,125/1,119 for ROTOWIRE and RW-FG respectively.

As for automatic evaluation, we use BLEU score (Papineni et al., 2002) and three extractive evaluation metrics RG (Relation Generation), CS (Content Selection), and CO (Content Ordering) (Wiseman et al., 2017). The main idea of the extractive metrics is to use an Information Extraction (IE) system to extract predicted records mentioned in the generated summary and compare them with records extracted from reference summary to evaluate the model. We denote r as the set of records extracted from the gold summary, and $\hat{r}$ as the set of records extracted from the generated summary. The RG measures the content fidelity by computing how many generated records in $\hat{r}$ can be find in the table. CS measures the ability of content selection by computing the precision, recall and F1 score between r and $\hat{r}$. CO measures the model's ability of organizing and ordering the selected records by computing the normalized Damerau-Levenshtein Distance between r and $\hat{r}$.

4.2 Implementation Details

We conduct our experiments on four 16GB NVIDIA V100 GPUs. We implement our generation model with FAIRSEQ (Ott et al., 2019). We use the pretrained BART (Lewis et al., 2020) to initialize the model. We use the Adam (Kingma and Ba, 2014) optimizer with learning rate 5e-5. The dropout rate is 0.1, weight decay is 0.1 and clip norm (Pascanu et al., 2013) is 0.1. We use the dynamic batching strategy with 2048 max tokens within a batch. We trained our model 100 epochs and select the best checkpoint at the validation set based on BLEU score. As for the XGBoost, we employ the available public library provided by (Chen and Guestrin, 2016). Since some tables have more rows, we pad features and labels in to fixed size with -1 and 0 respectively. At the decoding stage, the minimum decoding length is 330 and the maximum decoding length is 600 with n-gram-block (Paulus et al., 2018) to avoid 6-gram repetition.

Model	RG		CS			CO	BLEU
	P%	#	P%	R%	F1%		
ROTOWIRE							
Templ (Chen and Guestrin, 2016)	99.93	54.27	26.86	57.90	36.69	14.82	8.93
NCP (Puduppully et al., 2019a)	86.84	34.26	33.37	50.95	40.58	18.07	16.50
ENT (Puduppully et al., 2019b)	91.16	30.03	38.45	48.26	42.80	19.90	16.13
Hierarchical-k (Rebuffel et al., 2020)	88.46	41.82	33.07	48.60	39.35	17.79	16.77
Marco (Puduppully and Lapata, 2021)	**97.51**	41.09	35.57	**58.05**	44.11	19.87	15.46
HETD (Gong et al., 2019)	92.02	31.51	35.82	47.74	40.93	20.60	16.85
DUV (Gong et al., 2020)	86.27	26.97	40.44	48.61	44.15	23.14	15.92
HEnc (Li et al., 2021)†	93.14	32.73	40.80	55.88	47.16	25.30	17.96
BART	94.47	37.07	38.82	55.88	45.81	20.85	**19.13**
FLAP	93.21	21.39	**53.18**	46.24	**49.46**	**25.44**	18.32
RW-FG							
Templ (Chen and Guestrin, 2016)‡	98.89	51.80	23.98	43.96	31.03	10.25	12.09
ENT(Puduppully et al., 2019b)‡	93.72	35.69	39.04	49.29	43.57	17.5	21.23
NCP (Puduppully et al., 2019a)‡	94.21	35.99	43.31	55.15	48.52	23.46	23.86
NCP+TR (Wang, 2019a)‡	**95.70**	37.49	42.90	56.91	48.92	24.47	24.41
HEnc (Li et al., 2021)†	94.75	38.08	42.72	57.56	49.04	25.23	24.52
BART	95.03	37.41	44.89	**57.61**	50.46	24.92	**25.89**
FLAP	94.15	28.96	**53.58**	52.35	**52.95**	**26.16**	25.03

† HEnc (Li et al., 2021) doesn't released their source code as well as pretrained models and generated summaries, thus we use their results from the original paper.
‡ Results of Templ, ENT, NCP and NCP+TR on RW-FG are from Wang (2019a).

Table 1: Overall automatic metrics on the test set of ROTOWIRE and RW-FG dataset. The metrics include relation generation (RG), count (#), precision (P%), content selection (CS) precision (P%), re-call (R%) and content ordering (CO) in normalized Damerau-Levenshtein distance (DLD%)

4.3 Compared Baselines

We compare our system with several baselines: Templ (Chen and Guestrin, 2016), NCP+CC (Puduppully et al., 2019a), ENT (Puduppully et al., 2019b), Hierarachical-k (Rebuffel et al., 2020), Marco (Puduppully and Lapata, 2021), HETD (Gong et al., 2019), DUV (Gong et al., 2020) and HEnc (Li et al., 2021). We also use BART to train an end-to-end generation model that takes the entire table as input and summary as output without explicit content selection. To ensure pair comparison, we use the output summaries released by these papers or generate the corresponding summaries with their released checkpoints. We use the public available evaluation script and extraction models to compute automatic metrics.

4.4 Overall Results

We report the overall results on ROTOWIRE and RW-FG in Table 1. In ROTOWIRE, FLAP achieves the best performance on content selection precision, improving the previous state-of-the-art by 12.38% and raises the F1 score by 2.3%, which demonstrates that our model is superior at selecting appropriate and proper records. Our model also achieves the best score on content ordering (CO) and improves the previous best model by 0.14%, which shows that our model has a better ability to organize and resort the selected records. Additionally, although the two BART-based models achieve the best BLEU score, which demonstrates the powerful generation ability of the pretrained model, the performances of content selection and planning of the BART model still fall short in comparison with FLAP. This shows that it is difficult for the end-to-end BART to learn content selection and planning without explicit content selection supervision. What's more, notice that Marco gets the highest score on CS recall, but its precision is only 35.57, which is 17.61% lower than our model. This means that Marco just learns to mention as many records as possible, which is sub-optimal. Last but not least, FLAP has lower RG # than other models, which means that our model is more conservative when selecting the records. Overall speaking, FLAP achieves well-balanced and competitive per-

Test Set			
Model	**P%**	**R%**	**F1%**
SVM	38.95	39.70	39.32
Random Forest	**64.94**	37.73	47.73
XGBoost	58.58	**47.10**	**52.22**
Development Set			
SVM	39.51	39.89	39.70
Random Forest	**65.88**	38.55	48.64
XGBoost	59.64	**48.38**	**53.42**

Table 2: Performance of different classifier vs the ground truth label on ROTOWIRE.

formance in all metrics.

The results are similar in RW-FG. Since RW-FG is a cleaned and enrichment version of ROTOWIRE, all the baselines have higher performance on all the metrics. FLAP achieves consistent performance. Specifically, FLAP achieves the best performance on content selection precision F1 and CO again, and improves previous state-of-the-art by 10.86%, 3.91% and 0.93% respectively.

4.5 Performance of Different Content Selector

We report the performance of different classifiers compared with the ground truth 0-1 label in Table 2. We use SVM (Cortes and Vapnik, 1995), RandomForest (Ho, 1995) and XGBoost (Chen and Guestrin, 2016) respectively. As shown in the table, XGBoost and Random Forest achieve better performance than SVM since they are ensemble-based methods. Besides, XGBoost have higher performance on F1 and recall, specifically, it has 10% higher in recall and 5% higher in F1 than Random Forest. We think it may be because that XGBoost is a regularizing gradient boosting framework which can reduce variance, and also reduces bias. Notice that both XGBoost and Random Forest achieve higher F1 score than previous methods which treat table as sequence of token embeddings, which demonstrates the importance of scalar feature in content selection.

4.6 Effectiveness of Feature Indication and Dealing with Exposure Bias

In this subsection, we will discuss the effectiveness of the feature indication mechanism and study how feature indication can alleviate the exposure bias (Ranzato et al., 2016) problem. For this purpose, we implement a BART$_{pipe}$ model that takes the selected subset of table rather than the whole table as input. BART$_{pipe}$ can be seen to perform translation task that translate the selected records to natural language summary. In the training stage, we feed the model with the concatenation of ground truth mentioned records. In the evaluate stage, we feed the concatenation of records predicted by the content selector. We report the results of two models evaluated under different kinds of records and corresponded performance drop in Table 3.

From Table 3, first thing we can observe is that with the ground truth records, both BART$_{pipe}$ and FLAP achieves very high performance across all metrics. This can be viewed as an upper bound for the task. Given gold content selection, BART$_{pipe}$ performs better in terms of content selection and content ordering as it is strictly translating the selected records. While FLAP generates more records and has higher BLEU score as it learns explicit as well as implicit content selection, which provides more flexibility.

What's more, notice that both models suffer from performance drops when evaluated with predicted records. The performance of BART$_{pipe}$ in content selection, content ordering, and BLEU all drastically decrease. However, the drop of FLAP is much smaller, especially for BLEU score. We believe that the low BLEU of BART$_{pipe}$ is mainly caused by the combination distribution variation of the model input form rather than the information mismatch between ground truth and predicted records. The gap between FLAP and BART$_{pipe}$ given predicted records shows the necessity of whole table as input and that our feature indication mechanism does alleviate the exposure bias (Ranzato et al., 2016) problem.

We also report the results of the two models that trained on predicted records on Table 4. The training and evaluation process is consistent in that they both use predicted records. To avoid label leakage issue, we use 4-fold cross validation to get the predicted records on training set. As shown in Table 4, performance BART$_{pipe}$ trained on predicted records drops significantly. The reason is that it is difficult for a model to generate summary conditions on the predicted records along. Thus models trained on such training set tend to generate more ungrounded facts. Compared to BART$_{pipe}$, the performance of FLAP trained on predicted records doesn't drop sharply, because the input of FLAP is the entire table along with the feature indication sequence. Besides, although the training and evaluation process of this setting is consistent, the performance of FLAP trained on ground truth records

Test Set								
Model	**RDs**	**RG**		**CS**			**CO**	**BLEU**
		P%	**#**	**P%**	**R%**	**F1%**		
BART$_{pipe}$	gt	86.84	25.38	89.21	87.55	88.37	49.24	27.19
	pred	$88.12_{\uparrow 1.47\%}$	$22.39_{\downarrow \mathbf{11.74\%}}$	$50.84_{\downarrow 42.99\%}$	$42.66_{\downarrow 51.27\%}$	$46.39_{\downarrow 47.50\%}$	$23.51_{\downarrow 52.25\%}$	$15.17_{\downarrow 44.20\%}$
FLAP	gt	92.29	27.36	82.13	85.36	83.71	48.89	27. 54
	pred	$93.21_{\uparrow \mathbf{0.99\%}}$	$21.39_{\downarrow 21.82\%}$	$53.18_{\downarrow \mathbf{35.23\%}}$	$46.24_{\downarrow \mathbf{45.80\%}}$	$49.46_{\downarrow \mathbf{40.90\%}}$	$25.44_{\downarrow \mathbf{47.96\%}}$	$18.32_{\downarrow \mathbf{33.47\%}}$
Development Set								
BART$_{pipe}$	gt	86.93	24.80	89.30	87.74	89.00	48.81	26.53
	pred	$88.27_{\uparrow 1.54\%}$	$21.95_{\downarrow \mathbf{11.49\%}}$	$52.08_{\downarrow 41.67\%}$	$43.77_{\downarrow 50.11\%}$	$47.56_{\downarrow 46.56\%}$	$24.48_{\downarrow 49.84\%}$	$15.33_{\downarrow 42.21\%}$
FLAP	gt	92.81	26.72	82.13	85.36	83.48	48.44	27.34
	pred	$92.29_{\downarrow \mathbf{0.56\%}}$	$22.44_{\downarrow 18.37\%}$	$54.02_{\downarrow \mathbf{34.22\%}}$	$47.53_{\downarrow \mathbf{44.17\%}}$	$50.56_{\downarrow \mathbf{39.43\%}}$	$26.41_{\downarrow \mathbf{45.47}}$	$18.75_{\downarrow \mathbf{31.42\%}}$

Table 3: Results of BART$_{pipe}$ and FLAP on ROTOWIRE development and test set. For BART$_{pipe}$, gt means utilizing the ground truth records as input, pred means using predicted records. For FLAP, gt means utilizing the ground truth feature sequence as input, while pred means using the predicted feature sequence.

Test Set							
Model	**RG**		**CS**			**CO**	**BLEU**
	P%	**#**	**P%**	**R%**	**F1%**		
BART$_{pipe}$	50.74	19.16	28.01	37.92	32.22	16.03	15.35
FLAP	94.65	37.08	40.29	58.40	47.68	23.27	19.36
Development Set							
BART$_{pipe}$	50.58	18.98	28.11	39.07	32.69	15.86	15.31
FLAP	94.80	36.72	40.32	59.69	48.12	24.31	19.57

Table 4: Results of BART$_{pipe}$ and FLAP on ROTOWIRE that trained on predicted records and evaluated with predicted records.

is still better than that trained on predicted ones. This is because the predicted records can't align well with the summary. Thus FLAP trained on ground truth records has a better guidance. This once again demonstrates the effectiveness of our feature indication mechanism.

4.7 Ablation Study

Test Set							
Model	**RG**		**CS**			**CO**	**BLEU**
	P%	**#**	**P%**	**R%**	**F1%**		
baseline	94.47	37.07	38.82	55.88	45.81	20.85	19.13
+pt	**94.83**	36.98	39.80	**57.61**	47.07	22.89	**19.72**
+feat.inject	93.21	21.39	**53.18**	46.24	**49.46**	**25.44**	18.32
Development Set							
baseline	94.56	36.13	38.64	56.02	45.72	21.59	18.92
+pt	**94.87**	36.57	39.66	**58.12**	47.14	23.50	**19.72**
+feat.inject	92.29	22.44	**54.02**	47.53	**50.56**	**26.41**	18.75

Table 5: Automatic metric on on ROTOWIRE test set and development set. pt stands for numerical reasoning pretraining. feat.inject stands for feature indication.

In this subsection, we will discuss the effectiveness of feature indication and numerical reasoning pretraining. As shown in Table 5, the performance of baseline model improves greatly after adding numerical reasoning pretraining. Specifically, content selection precision recall, F1 and content ordering increase around 2% on both test and development set. The BLEU score also increases. During pretraining, the model acquire the ability of understanding and reasoning over tables, thus pt not only improves the performance on content selection, but also improves the performance on content planning. Second, after adding feature indication, the performance of content selection precision largely increased by around 14% on both test and development set. CO increased by around 3%. It demonstrates that the auxiliary can remarkably improve the content selection performance and then help content planning.

We also showcase a couple of case studies in the appendix A to demonstrate how FLAP can generate well-balanced summary than end-to-end BART and pipeline BART and an analysis on minimum decoding length in appendix B.

5 Conclusions

In this work, we present an end-to-end model that takes whole table as input to alleviate the exposure bias problem. We propose a feature indication mechanism and utilizes the table's scalar feature and an XGBoost as content selector. We also propose a numerical reasoning pretraining task that can improve the performance of content planning. Experiments on two document level table-to-text datasets ROTOWIRE and RW-FG show that our model achieves favorable performance over several strong baselines. We also propose experiments to demonstrate the effectiveness of each component of our model. We hope the idea of treating the values in the table as numerical features and the proposed numerical reasoning pretraining can inspire future work on table-to-text generation.

References

Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 785–794. ACM.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

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.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *J. Artif. Intell. Res.*, 61:65–170.

Heng Gong, Wei Bi, Xiaocheng Feng, Bing Qin, Xiaojiang Liu, and Ting Liu. 2020. Enhancing content planning for table-to-text generation with data understanding and verification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 2905–2914. Association for Computational Linguistics.

Heng Gong, Xiaocheng Feng, Bing Qin, and Ting Liu. 2019. Table-to-text generation with effective hierarchical encoder on three dimensions (row, column and time). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3143–3152.

Barbara J Grosz, Aravind K Joshi, and Scott Weinstein. 1994. Centering: A framework for modelling the coherence of discourse.

Tin Kam Ho. 1995. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

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, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. Association for Computational Linguistics.

Liang Li, Can Ma, Yinliang Yue, and Dayong Hu. 2021. Improving encoder by auxiliary supervision tasks for table-to-text generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5979–5989, Online. Association for Computational Linguistics.

William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text-interdisciplinary Journal for the Study of Discourse*, 8(3):243–281.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. Fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53.

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, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318. ACL.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318. PMLR.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

Laura Perez-Beltrachini and Mirella Lapata. 2018. Bootstrapping generators from noisy data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1516–1527. Association for Computational Linguistics.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019a. 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.

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

Ratish Puduppully and Mirella Lapata. 2021. Data-to-text generation with macro planning. *Trans. Assoc. Comput. Linguistics*, 9:510–527.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Clément Rebuffel, Laure Soulier, Geoffrey Scoutheeten, and Patrick Gallinari. 2020. A hierarchical model for data-to-text generation. In *Advances in Information Retrieval - 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14-17, 2020, Proceedings, Part I*, volume 12035 of *Lecture Notes in Computer Science*, pages 65–80. Springer.

Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87.

Richard S Sutton, Andrew G Barto, et al. 1998. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge.

Lisa Torrey and Jude Shavlik. 2010. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Hongmin Wang. 2019a. Revisiting challenges in data-to-text generation with fact grounding. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 311–322, Tokyo, Japan. Association for Computational Linguistics.

Hongmin Wang. 2019b. Revisiting challenges in data-to-text generation with fact grounding. In *Proceedings of the 12th International Conference on Natural Language Generation, INLG 2019, Tokyo, Japan, October 29 - November 1, 2019*, pages 311–322. Association for Computational Linguistics.

Sam Wiseman, Stuart M. Shieber, and Alexander M. Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2253–2263. Association for Computational Linguistics.

A Case Study

Figure 5 shows an example of the records extracted from summaries generated by FLAP and BART. We can find that BART tends to mention more records regardless of importance. Such as (Henson, 0, PLAYER-FG3M) and (Henson, 0, PLAYER-OREB). While FLAP can capture important records, like (Quincy, 18, PLAYER-PTS), and neglect unnecessary ones. We attributed this phenomenon to the lack of explicit selection procedure of BART. This demonstrates that even a powerful pretrained language model can't learn content selection well without explicit content selection supervision and the advantage of our feature indication mechanism.

FLAP
... (Davis,42,PLAYER-MIN),(Omer,16,PLAYER-PTS), (Omer,11,PLAYER-REB),(Tyreke,12,PLAYER-PTS), (Tyreke,nine,PLAYER-REB),(Tyreke,six,PLAYER-AST), (Quincy,18,PLAYER-PTS),(Giannis,15,PLAYER-PTS), (Giannis,nine,PLAYER-REB),(Ersan,15,PLAYER-PTS), (Henson,14,PLAYER-PTS),(Bayless,14,PLAYER-PTS)
BART
... (Bucks,37,TEAM-FG_PCT),(Bucks,20,TEAM-FG3_PCT), (Davis,6,PLAYER-FGM),(Davis,18,PLAYER-FGA), (Davis,8,PLAYER-FTM),(Davis,9,PLAYER-FTA), (Henson,one,PLAYER-STL),(Zaza,11,PLAYER-PTS), (Zaza,3,PLAYER-FGM),(Zaza,9,PLAYER-FGA), (Zaza,5,PLAYER-FTM),(Zaza,6,PLAYER-FTA), (Zaza,11,PLAYER-REB),(Henson,14,PLAYER-PTS), (Henson,7,PLAYER-FGM),(Henson,10,PLAYER-FGA), (Henson,0,PLAYER-FG3M),(Henson,4,PLAYER-FG3A), (Henson,0,PLAYER-FTM),(Henson,0,PLAYER-TO), (Henson,6,PLAYER-FTA),(Henson,seven,PLAYER-REB), (Henson,0,PLAYER-OREB),(Henson,0,PLAYER-PF)

Figure 5: Example of records extracted from the generated summaries of FLAP and BART. Common records are removed. Correct records are lighted with green, redundant ones are lighted with orange.

Figure 6 shows an example of generated summaries of FLAP and BART$_{pipe}$ condition on the same predicted records of XGBoost. As shown on the Figure, BART$_{pipe}$ generates sentences contradict to facts. For example, "...as he totaled 7 points and 7 rebounds in the victory..." is contradict to the fact that "The Milwaukee Bucks (18-17) defeated the New York Knicks (5-31) 95-82...". While FLAP can generate fluent and consistent summary.

FLAP
The Milwaukee Bucks (18-17) defeated the New York Knicks (5-31) 95-82 on Tuesday. …The Knicks played this game without a true star, as Carmelo Anthony missed this game with a shoulder injury. Tim Hardaway Jr. led the team with 17 points, while J.R. Smith accrued 15 points, seven rebounds and four assists of his own. Cole Aldrich was the only other starter in double figures, as he scored 12 points…
BART$_{pipe}$
The Milwaukee Bucks (18-17) defeated the New York Knicks (5-31) 95-82 on Saturday. …J.R. Smith was fantastic, as he tallied 15 points, 7 assists and 7 rebounds in the win. Tim Hardaway Jr. continued his scoring tear off the bench, as he scored 17 points, while filling in for Lance Thomas. Quincy Acy had the most production off the bench, as he totaled 7 points and 7 rebounds in the victory. Cole Aldrich led the team in scoring, as he dropped 12 points off the bench…

Figure 6: Example of generated summaries of FLAP and BART$_{pipe}$ condition on the same predicted records of XGBoost.

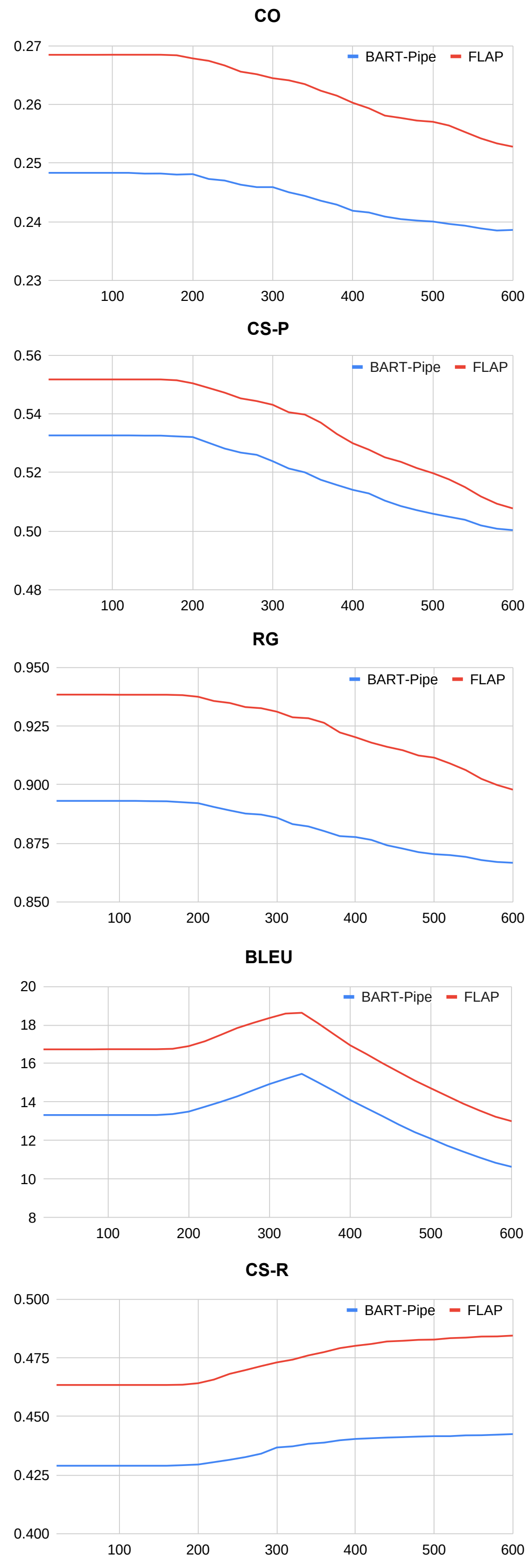

Figure 7: Automatic metric of a trained model under different minimum decoding length on ROTOWIRE development set.

B Discussion on the Effect of Minimum Decoding Length

In this subsection, we will discuss the effect of minimum decoding length to each metric. As shown in Figure 7, the first thing to noticed is that RG, CS-P and CO decrease as minimum decoding length increases. This is because as the model generates more tokens, it is more likely to generate wrong records that will punish precision score. Secondly, content selection recall grows up as minimum decoding length increase, because as the model generates more tokens, more likely a target record will be mentioned. And BLEU goes up until an inflection point. The reason may be that model can improve the BLEU score by generating high-frequency n-grams until the length penalty mechanism of BLEU score starts to work. However, these n-grams are usually dull and redundant. Lastly, we can observe that our model can consistently outperform the BART$_{pipe}$ model by a large margin, which demonstrates the effectiveness of the feature indication mechanism and numerical reasoning pretraining again. This phenomenon also reminds us that we should not focus too much on BLEU and CS-R in the document level table-to-text generation task since these two metrics can be cheated by forcing the model to output more tokens.