SYSTRAN @ WNGT 2019: DGT Task

Li Gong, Josep Crego, Jean Senellart
SYSTRAN / 5 rue Feydeau, 75002 Paris, France
`firstname.lastname@systrangroup.com`

Abstract

This paper describes SYSTRAN participation to the Document-level Generation and Translation (DGT) Shared Task of the 3rd Workshop on Neural Generation and Translation (WNGT 2019). We participate for the first time using a Transformer network enhanced with modified input embeddings and optimising an additional objective function that considers content selection. The network takes in structured data of basketball games and outputs a summary of the game in natural language.

1 Introduction

Data-to-text generation is an important task in natural language generation (NLG). It refers to the task of automatically producing a descriptive text from non-linguistic structured data (tables, database records, spreadsheets, *etc.*). Table 1 illustrates an example of data-to-text NLG, with statistics of a NBA basketball game (top) and the corresponding game summary (bottom).

Different from other NLG tasks (e.g., machine translation), data-to-text generation faces several additional challenges: First, data-to-text generation models have to select the content before generating text. In machine translation, the source and target sentences are semantically equivalent to each other, whereas in data-to-text generation, the model initially selects appropriate content from the input data to secondly generate fluent sentences that incorporate the selected content; Second, the training data in data-to-text generation task is often very limited. Unlike machine translation, where training data consist of translated sentence pairs, data-to-text generation models are trained from examples composed of structured data and its corresponding descriptive summary, which are much harder to produce.

In this paper, we tackle both challenges previously discussed. We introduce a new data-to-text generation model which jointly learns content selection and text generation. In addition, we present two data augmentation methods that further boost performance of the NLG system.

2 Data Resources

We use the official data set made available for the WNGT 2019 DGT task (Hayashi et al., 2019). It consists of the the ROTOWIRE dataset (Wiseman et al., 2017), a dataset of NBA basketball game summaries, paired with their corresponding box- and line-score tables. Table 1 illustrates an example of the dataset. In the box-score table, each team has at most 13 players and each player is described by 23 types of values. In the line-score table, each team has 15 different types of values. As for the associate summaries, the average length is 337 tokens, and the vocabulary size is $11.3K$. The ROTOWIRE dataset contains $4,853$ summaries in total, in which $3,398$ summaries are for training, 727 for validation and 728 for test. In addition, the next monolingual resources were considered usable in all tracks:

- Any monolingual data allowable by the WMT 2019 English-German news task,
- Pre-trained word/subword/character embeddings (e.g., GloVe, fastText),
- Pre-trained contextualized embeddings (e.g., ELMo, BERT),
- Pre-trained language models (e.g., GPT-2).

3 Data-to-Text Transformer Model

In this section, we present how we adapt the Transformer model for the data-to-text generation tasks. First, the input embedding of Transformer encoder is replaced by our record embedding to better incorporate the record information. Second, a new

TEAM-NAME	WIN	LOSS	PTS	AST	...
Cavaliers	51	28	90	25	...
Celtics	37	42	99	30	...

NAME	POS	PTS	FGM	FGA	CITY	...
LeBron James	F	14	5	14	Cleveland	...
Kevin Love	F	19	6	12	Cleveland	...
Kyrie Irving	N/A	N/A	N/A	N/A	Cleveland	...
Brandon Bass	F	12	6	8	Boston	...
Avery Bradley	G	15	7	12	Boston	...
Marcus Smart	G	19	7	10	Boston	...
...	...	...	...	...	...	...

POS: position, PTS: points, FGM: Player field goals made; FGA: Player field goals attempted; AST: assists; CITY: player team city.

The **Boston Celtics** (**37**-**42**) defeated the **Cleveland Cavaliers** (**51**-**28**) **99**-**90** on Friday in **Cleveland**. With the **Cavaliers** solidified as the No.2 seed for the Eastern Conference playoffs , they did not try particularly hard to win this game, starting with sitting **Kyrie Irving** (hip) to make sure he stays healthy. Regardless, the **Celtics** took advantage, picking up a huge victory as they fight to stay in the playoffs. **Marcus Smart** led the way scoring for **Boston**, posting **19** points on **7**-of-**10** shooting in **34** minutes. **Avery Bradley** was close behind, scoring **15** points (**7**-**12** FG) in **31** minutes. The **Celtics** starting frontcourt of **Tyler Zeller** and **Brandon Bass** combined to score 25 points and grab 11 rebounds. **Boston**'s final starter, **Evan Turner**, struggled shooting the ball and only scored **four** points, but still managed to dish out **13** assists and grab **six** rebounds. **Isaiah Thomas**, as he has done so well since joining the **Celtics**, provided solid production off the bench, scoring **17** points (**4**-**12** FG, **2**-**6** 3Pt) in **23** minutes. ...

Table 1: Example of data-records (left) and document summary (right) from the ROTOWIRE dataset. Entities and values corresponding to data-records are boldfaced.

learning objective is added into our model to improve its content-oriented performance.

3.1 Record Embedding

The input of data-to-text model encoder is a sequence of records. Each record is a tuple of four features (*Entity*, *Type*, *Value*, *Info*). Inspired by previous work (Yang et al., 2016; Wiseman et al., 2017; Puduppully et al., 2019), we embed features into vectors, and use the concatenation of feature embeddings as the embedding of record.

$$\mathbf{r}_i = [\mathbf{r}_{i,1}; \mathbf{r}_{i,2}; \mathbf{r}_{i,3}; \mathbf{r}_{i,4}] \quad (1)$$

where $\mathbf{r}_i \in \mathbb{R}^{dim}$ is the ith record embedding in the input sequence and $\mathbf{r}_{i,j} \in \mathbb{R}^{\frac{dim}{4}}$ is the jth feature embedding in $\mathbf{r}_i$.

Since there is no order relationship within the records, the positional embedding of the Transformer encoder is removed.

3.2 Content Selection Modeling

Besides record embedding, we also add a new learning objective into the Transformer model.

As presented before, we need to select the content from the input records before generating the output summary. Some records are generally important no mater the game context, such as the team name record and team score record, whereas the importance of some other records depend on the game context. For example, a player having the highest points in the game is more likely to be mentioned in the game summary. Within the Transformer architecture, the self-attention mechanism can generate the latent representation for each record by jointly conditioning on all other records in the input dataset. A binary prediction layer is added on top of the Transformer encoder output (as shown in Figure 1) to predict whether or not one record will be mentioned in the target summary.

The architecture of our data-to-text Transformer model is shown in Figure 1. As presented before, the encoder takes the record embedding as input and generates the latent representation for each record in the input sequence. The output of encoder is then used to predict the importance of each record and also serves as the context of the decoder. The decoder of our model is the same as the original Transformer model in machine translation. It predicts the next word conditioned on the encoder output and the previous tokens in the summary sequence.

In content selection modeling, the input record sequences together with its label sequences are used to optimize the encoder by minimizing the cross-entorpy loss. In language generation training, the encoder and decoder are trained together to maximize the log-likelihood of the training data. The two learning objectives are trained alternatively.

4 Data Augmentation Methods

In data-to-text generation task, the model needs to not only generate fluent text, but also generate text which is coherent with the input records. Several content-oriented evaluation metrics are proposed in (Wiseman et al., 2017) to evaluate such cohesion, including the precision of record generation and the recall rate with respect to the records in gold summary.

In this section, we present two data augmentation methods: *synthetic data generation* and *train-*

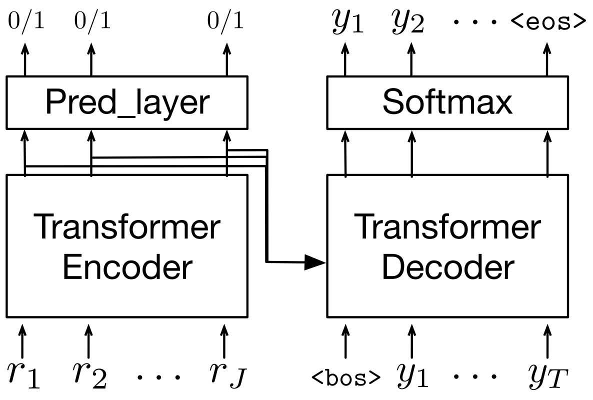
0/1
0/1
0/1
Pred_layer
Transformer Encoder
r_1 r_2 $\cdots$ r_J
y_1 y_2 $\cdots$ <eos>
Softmax
Transformer Decoder
<bos> y_1 $\cdots$ y_T

Figure 1: Model Architecture

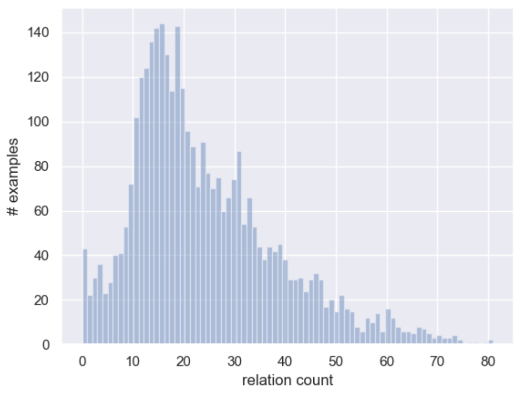

Figure 2: relation count distribution in training data.

ing data selection. Each of them has different impacts on the content-oriented evaluation results.

4.1 Synthetic Data Generation

In order to improve the cohesion between the input records and output summary, we need more data to enhance the encoder-decoder attention of the decoder. Here we introduce a method to generate synthetic training data.

We first randomly change the values of records and the changed record set ($\mathbf{s}'$) is then used to generate automatic summary ($\mathbf{t}'$) by a trained data-to-text system. The synthetic data pairs ($\mathbf{s}'$, $\mathbf{t}'$) are then used to improve such system.

In order to keep the data cohesion in the table, the change is constrained with the following rules:

- only numeric values are changed. Non-numeric values such as the position of a player or the city name of a team are kept the same.
- after the change, the team scores should not violate the win/loss relation
- the changed values should stay in the normal range of its value type. It should not bigger than its maximum value or smaller than its minimum value through all games.

Our data generation technique doubles the amount of training data available for learning.

4.2 Training Data Selection

A deficiency of data-to-text NLG systems is the poor coverage of relations produced in the generated summaries. In order to increase the coverage, a simple solution consists of learning to produce a larger number of relations. Here, we present a straightforward method to bias our model to output more relations by means of fine-tuning on the training examples containing a greater number of relations.

We use an information extraction (IE) system to extract the number of relations of each training summary. Then, we select for fine-tuning our baseline model the subset of training data in which each summary contains at least N relations. In this work, we take advantage of the IE system[1] provided by (Puduppully et al., 2019), and the distribution of the number of relations in the training summary is illustrated in Figure 2.

5 Experimental Setup

5.1 Data and Preprocessing

We run the experiments with the ROTOWIRE dataset (Wiseman et al., 2017), a dataset of NBA basketball game summaries, paired with their corresponding box- and line-score tables. Table 1 illustrates an example of the dataset. In the box-score table, each team has at most 13 players and each player is described by 23 types of values. In the line-score table, each team has 15 different types of values. In addition, the date of each game is converted into the day of the week (such as "*Saturday*") as an additional record. In the pre-processing step, the input box- and line-score tables are converted into a fix-length sequence of records. Each sequence contains 629 records.[2] As for the associate summaries, the average length is 337 tokens, and the vocabulary size is 11.3K. The

[1]The model is publicly available at `https://github.com/ratishsp/data2text-plan-py`

[2]In the 629 records, 598 records are for players, 30 records for teams and 1 record for the date.

ROTOWIRE dataset contains 4853 summaries in total, in which 3398 summaries are for training, 727 for validation and 728 for test.

In content selection modelling, we need the labels of input records to indicate which records in the input will be mentioned in the output summary. Here we use a very simple method to generate such labels. First, we label the entity records[3]. An entity record is labeled as 1 if its value is mentioned in the associated summary, otherwise it is labeled as 0. Second, for each player or team mentioned in the summary, the rest of its values in the table are labeled as 1 if they occur in the same sentence in the summary.

5.2 Evaluation metrics

The model output is evaluated with BLEU (Papineni et al., 2002) as well as several content-oriented metrics proposed by (Wiseman et al., 2017) including three following aspects:

- Relation Generation (RG) evaluates the number of extracted relations in automatic summaries and their correctness (precision) w.r.t the input record dataset;
- Content Selection (CS) evaluates the precision and recall rate of extracted relations in automatic summaries w.r.t that in the gold summaries;
- Content Ordering (CO) evaluates the normalized Damerau-Levenshtein Distance (Brill and Moore, 2000) between the sequence of extracted relations in automatic summaries and that in the gold summaries.

All these content-oriented metrics are based on an IE system which extracts record relations from summaries. For the purpose of comparison, we directly use the publicly available IE system of (Puduppully et al., 2019) to evaluate our models.

5.3 Training Details

In all experiments, we use our model with 1 encoder layer and 6 decoder layers, 512 hidden units (hence, the record feature embedding size is 128, see Section 3), 8 heads, GELU activations (Hendrycks and Gimpel, 2016), a dropout rate of 0.1 and learned positional embedding for the decoder. The model is trained with the Adam optimizer (Kingma and Ba, 2014), learning rate is fixed to 10^{-4} and batch size is 6. As for inference, we use beam size 4 for all experiments, and the maximum decoding length is 600.

We implement all our models in Pytorch, and train them on 1 GTX 1080 GPU.

Model	RG		CS		CO	BLEU
	#	P%	P%	R%	DLD%	
GOLD	23.32	94.77	100	100	100	100
TEMPL	54.29	99.92	26.61	59.16	14.42	8.51
WS-2017	23.95	75.10	28.11	35.86	15.33	14.57
NCP-2019	33.88	87.51	33.52	51.21	18.57	16.19
DATA-TRANS	23.31	79.81	36.90	43.06	22.75	20.60
+DATA_GEN	22.59	**82.49**	**39.48**	42.84	23.32	19.76
+DATA_SEL	**26.94**	79.54	35.27	**47.49**	22.22	19.97
+BOTH	24.24	80.52	37.33	44.66	23.04	20.22

Table 2: Automatic evaluation on ROTOWIRE development set using relation generation (RG) count (#) and precision (P%), content selection (CS) precision (P%) and recall (R%), content ordering (CO) in normalized Damerau-Levenshtein distance (DLD%), and BLEU.

6 Results

The results of our model on the development set are summarized in Table 2. GOLD represents the evaluation result on the gold summary. The RG precision rate is 94.77%, indicating that the IE system for evaluation is not perfect but has very high precision. After that, results of three contrast systems are reported, where TEMPL and WS-2017 are the updated results[4] of Wiseman et al. (2017) models. TEMPL is template-based generator model which generates a summary consisting of 8 sentences: a general description sentence about the teams playing in the game, 6 player-specific sentences and a conclusion sentence. WS-2017 reports an encoder-decoder model with conditional copy mechanism. NCP-2019 is the best system configuration (NCP+CC) reported in (Puduppully et al., 2019) which is a neural content planning model enhanced with conditional copy mechanism. As for our model, results with four configurations are reported.

DATA-TRANS represents our data-to-text Transformer model (as illustrated in Figure 1) without any data augmentation. Comparing to NCP-2019, our model performs 3.4% higher on content selection precision, 4.2% higher on

[3]Record whose *Value* feature is an entity (see Section **??**), for example: "LeBron_James|NAME|LeBron_James|H/W". The labeling is according to the *Value* feature

[4]Here we all use the IE system of (Puduppully et al., 2019) which is improved from the original IE system of (Wiseman et al., 2017)

content ordering metric and 4.4 points higher on BLEU. Our model performs better on the CO metric, we attribute this improvement to that our model generates nearly the same number of relations as the gold summary which reduces the edit distance between the two sequences of relations. However, our model is 7.7% lower on RG precision. And on the CS recall rate, our model is 8.2% lower than NCP-2019. This is probably due to the fact that NCP-2019 generates much more records than our model (33.88 vs. 23.31) which could result higher coverage on the relations in gold summary.

Comparing to TEMPL and WS-2017, our model is much better on BLEU and CS precision. Our model generates nearly the same number of relations as WS-2017, but with 7.2% higher on recall rate and 7.4% higher on CO metric.

By synthetic data generation (+DATA_GEN), we generate synthetic table records as described in secion 4.1. These synthetic table records are then used as input to the DATA-TRANS model to generate summaries. All training table records are used to generate synthetic data. The synthetic data is then combined with the original training data to fine-tune the DATA-TRANS model. From Table 2, we can see that the RG and CS precisions are both improved by 2.7% and 2.6% respectively. There is no significant change on others metrics. The CO metric is slightly improved due to higher RG and CS precisions. The CS recall rate is slightly degraded with the number of extracted relations.

By training data selection (+DATA_SEL), we select the data whose summary contains the number of relations $N >= 16$ as the new training data. The result training data size is 2242 (original size: 3398). It is then used to fine-tune the DATA-TRANS model. As shown in Table 2, as expected, the model after fine-tuning generates more relations in the output summaries. The average number of relations in the output summaries increases from 23.31 to 26.94. Respectively, the CS recall is increased from 43.06% to 47.49%. However, the CS precision is slightly degraded by 1.6%.

Finally, we combine both of the data augmentation methods (+BOTH). Synthetic data generation improves the RG and CS precisions. Training data selection improves the CS recall rate by making the model generate more relations. To combine the two methods, we choose to fine-tune the +DATA_GEN model with the selected training data of +DATA_SEL (so this configuration is actually +DATA_GEN+DATA_SEL). As shown in Table 2, all content-oriented evaluation metrics are improved compared to DATA-TRANS but not as much as each single of the data augmentation method. This configuration is like a trade-off between the two data augmentation configurations.

Model	RG		CS		CO	BLEU
	#	P%	P%	R%	DLD%	
TEMPL	54.23	99.94	26.99	58.16	14.92	8.46
WS-2017	23.72	74.80	29.49	36.18	15.42	14.19
NCP-2019	34.28	87.47	34.18	51.22	18.58	16.50
DATA-TRANS	24.12	79.17	36.48	42.74	22.40	20.16
+DATA_GEN	24.01	**83.89**	**38.98**	42.85	23.02	19.48
+DATA_SEL	**27.47**	80.70	35.33	**46.25**	21.87	20.03
+BOTH	24.80	81.08	37.10	43.78	22.51	20.14

Table 3: Automatic evaluation on ROTOWIRE test set.

Results on the test set are reported in Table 3. They follow the same pattern as those found on the development set. Our DATA-TRANS model outperforms all other contrast systems on BLEU, CS precision and content ordering metrics. The synthetic data generation method helps to improve the RG and CS precisions. The training data selection method improves the CS recall by making the model generate more relations. Combining these two data augmentation methods, all content-oriented evaluation results are improved compared to DATA-TRANS. However, there is no significant change on BLEU.

7 Conclusions

We present an enhanced Transformer-based data-to-text generation model for the WNGT2019 English NLG task. Experimental results have shown that our enhanced transformer model outperforms current state-of-the-art system on BLEU, content selection precision and content ordering metics. In addition, we proposed two data augmentation methods, each of them improves different content-oriented evaluation metrics.

References

Eric Brill and Robert C Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 286–293. Association for Computational Linguistics.

Hiroaki Hayashi, Yusuke Oda, Alexandra Birch, Ioannis Constas, Andrew Finch, Minh-Thang Luong, Graham Neubig, and Katsuhito Sudoh. 2019. Findings of the third workshop on neural generation and translation. In *Proceedings of the Third Workshop on Neural Generation and Translation*.

Dan Hendrycks and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units.

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

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 on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-text generation with content selection and planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6908–6915.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2017. Challenges in data-to-document generation. *arXiv preprint arXiv:1707.08052*.

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2016. Reference-aware language models. *arXiv preprint arXiv:1611.01628*.