An Encoder with non-Sequential Dependency for Neural Data-to-Text Generation

Feng Nie[1*] **Jinpeng Wang**[2] **Rong Pan**[1] **Chin-Yew Lin**[2]
[1]Sun Yat-Sen University [2]Microsoft Research Asia
[1]fengniesysu@gmail.com, [1]panr@sysu.edu.cn
[2]{jinpwa, cyl}@microsoft.com

Abstract

Data-to-text generation aims to generate descriptions given a structured input data (i.e., a table with multiple records). Existing neural methods for encoding input data can be divided into two categories: a) pooling based encoders which ignore dependencies between input records or b) recurrent encoders which model only sequential dependencies between input records. In our investigation, although the recurrent encoder generally outperforms the pooling based encoder by learning the sequential dependencies, it is sensitive to the order of the input records (i.e., performance decreases when injecting the random shuffling noise over input data). To overcome this problem, we propose to adopt the self-attention mechanism to learn dependencies between arbitrary input records. Experimental results show the proposed method achieves comparable results and remains stable under random shuffling over input data.

1 Introduction

Data-to-text generation, one classic task of natural language generation, aims to produce a piece of texts that adequately and fluently describes its structured input data (i.e., tables) (Kukich, 1983; Reiter and Dale, 1997; Barzilay and Lapata, 2005; Angeli et al., 2010; Kim and Mooney, 2010; Perez-Beltrachini and Gardent, 2017). Traditionally, it is divided into two subtasks: content selection (i.e., *what to say*) and the surface realization (i.e., *how to say*) (Reiter and Dale, 1997; Gatt and Krahmer, 2018). Recent neural generation systems ignore the distinction of these two subtasks using a single encoder-decoder model (Sutskever et al., 2014) with attention mechanism (Bahdanau et al., 2015; Mei et al., 2016; Dušek and Jurcicek, 2016; Kiddon et al., 2016; Chisholm et al., 2017).

[*]Contribution during internship at Microsoft.

Input	
Birth name	Johnny Allen Hendrix
Born	November 27, 1942 Seattle, Washington, U.S.
Genres	Rock, psychedelic rock, hard rock, blues, R&B
Occupation	Musician, songwriter, producer
Instruments	Guitar, vocals
Reference	
James Marshall Hendrix ... was an American rock guitarist, singer and songwriter.	

Table 1: An example of generating descriptions from the input data.

The encoder-decoder architecture first encodes the input data (e.g., a table) into a dense representation, where the input contains a set of records. Then descriptions are produced based on the input representation. The appropriate encoding method of input structured data remains an open question. Existing encoding methods for an input table can be decomposed into two stages: 1) converting each record in the table to a record vector, 2) combining the record vectors using a pooling method or a recurrent neural network (RNN) to represent the input table. In this paper, we investigate these two types of neural encoding methods over several data-to-text datasets. The empirical results show that RNN based methods outperform simple pooling methods in terms of BLEU evaluation.

The major difference between pooling and RNN based methods lies in the fact that pooling methods treat records in the input table independently while RNN based methods model the relationships among the records by treating the input records as a sequence. As a result, it is common that two records in the input data are relevant. For example, as shown in Table 1, the input record "Instruments: Guitar, vocals" is related to "Occupation: Musician, songwriter, producer".

The improvements of RNN based methods over

Proceedings of The 12th International Conference on Natural Language Generation, pages 141–146,
Tokyo, Japan, 28 Oct - 1 Nov, 2019.

pooling methods suggest that capturing dependencies among the input records is helpful. However, RNN based methods capture only the sequential relationships among the input data, which is sensitive to its input order. Given an input table, intuitively, permutations over the records should make no change to input representations, while we observe large performance decrease of RNN based methods when injecting the random shuffling noise over input data. To address this undesired nature of RNN, we propose using a self-attention mechanism to capture the dependency and enable the encoding to be less sensitive to any permutation noise. The experimental results on several datasets show self-attention based encoder achieves comparable results than RNN based methods and is more robust handling the input shuffling noise.

2 Method

The neural data-to-text generation is based on the encoder-decoder architecture. As shown in Figure 1 , there are multiple choices of table encoding that affect the generation decoder. We briefly introduce the backbone of the neural generation method in Section 2.1 and then introduce the details of three types of table encoders in Section 2.2.

2.1 Base Model

Given a set of records $S = \{r_j\}_{j=1}^{K}$, the goal of data-to-text generation is to produce a description $y = y_1, ..., y_T$. Usually, the encoder-decoder architecture consist of a table encoder and a recurrent neural network based decoder segmented with attention (Bahdanau et al., 2015) and conditional copy (See et al., 2017) mechanism. Firstly, each input record r_j is encoded into a hidden vector $\mathbf{h}_j$ using a specified table encoder, which is the focus of this paper and three encoders will be introduced in Section 2.2. Then, for the generated description y, the decoder generates the word y_t at the t-th time step based on the previously generated words $y_{<t}$ and the input hidden vectors $\mathbf{H} = \{\mathbf{h}_j\}_{j=1}^{K}$. Specifically,

$$P(y_t|y_{<t}, \mathbf{H}) = \text{softmax}(f(\mathbf{d}_t, y_{t-1}, \mathbf{c}_t)) \quad (1)$$

where $f(.)$ is a tanh function and $\mathbf{d}_t = \text{LSTM}(\mathbf{d}_{t-1}, y_{t-1}, , \mathbf{c}_{t-1})$ is the hidden state of the decoder at step t. $\mathbf{c}_t$ in Eq. 1 is the context vector at timestep t, computed as a weighted sum of input hidden vectors $\mathbf{h}_j$:

$$\mathbf{c}_t = \sum_{j=1}^{K} \alpha_{t,j} \mathbf{h}_j \quad (2)$$

where we use the attention model introduced in (Bahdanau et al., 2015) to compute the attention weight $\alpha_{t,j}$.

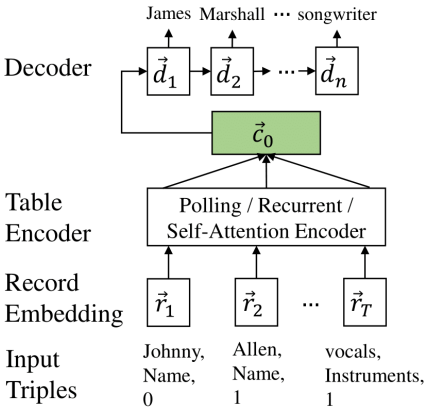

Figure 1: Overview of encoder-decoder architecture with different encoding methods.

2.2 Table Encoder

Record Vectors: The input table can be viewed as a set of field-value records, where values are sequences of words corresponding to a certain field (Liang et al., 2009; Lebret et al., 2016; Yang et al., 2017). For instance, in the Table 1, the word "William" has the field "Birth name" and it is the first word in this field. Every word in the field is a record r and is presented as triple (r^v, r^f, r^{pos}), where r^v, r^f and r^{pos} refer to the value (e.g., William), the field name (e.g., Birth name), the relative position in its field (e.g., 0). We map each record $r \in S$ into a vector $\mathbf{r}$ by concatenating the embedding of r^v, r^f and r^{pos}, denoted as $\mathbf{r} = [\mathbf{e}^v, \mathbf{e}^f, \mathbf{e}^{pos}]$, where $\mathbf{e}^v$, $\mathbf{e}^f$, $\mathbf{e}^{pos}$ are trainable word embeddings of r^v, r^f and r^{pos}.

Pooling Based Encoders: The pooling based encoder treats input records independently, therefore, it first applies a feed forward neural network layer over every record vector $\mathbf{r}_j$ and yields the input hidden vector $\mathbf{h}_j = \tanh(W\mathbf{r}_j)$, where W is a trainable parameter. The initial context vector $\mathbf{c}_0$ in Eq. 1 is calculated by the following max-pooling layer.

$$\mathbf{c}_0 = \max([\mathbf{h}_0, ..., \mathbf{h}_T]^T) \quad (3)$$

Recurrent Encoders: Different from pooling based encoder, the recurrent encoder captures the

	E2E	WIKIBIO
#Train	42061	582695
#Validation	4672	72831
#Test	4693	72831
Average field number	5.4	19.7
Average input length	20.1	53.1

Table 2: Statistics of two datasets.

dependency among the records by treating the set of record vectors $\mathbf{r}_1, ..., \mathbf{r}_T$ as a sequence. The sequence of records are fed into a RNN yielding a sequence of input hidden vectors $\mathbf{h}_1, ...\mathbf{h}_T$. We adopt a bidirectional LSTM by following (Mei et al., 2016). The initial context vector is set as the last hidden vector of the sequence $\mathbf{c}_0 = \mathbf{h}_T$.

Self-Attention Encoders: For data-to-text generation, input records are order invariant as input data should convey the same information regardless of the order of input records. The input records is a set, while the recurrent encoder makes strong hypothesis and treats it as a sequence.

Therefore an ideal table encoder has two desired properties: a) enable to capture relationships among the input records and b) is also order invariant. Recently proposed self-attention mechanism (Vaswani et al.; Wang et al., 2017) is able to learn interactions between arbitrary records and therefore is also irrelevant to the order of the records. For this purpose, we adapt the multi-layer self-attention mechanism for the encoding. Each layer has two sub-layers: one layer is for multi-head self-attention and the other one is a position wise feed-forward neural network with layer normalization (Vaswani et al.). Specifically,

$$s_{i,j} = \frac{Q_i^l \mathbf{h}_i^l (K_j^l \mathbf{h}_j^l)^T}{\sqrt{d_k}}; \beta_{i,j} = \frac{e^{s_{i,j}}}{\sum_{n=1}^{T} e^{s_{i,n}}} \quad (4)$$

$$\mathbf{h}_i^l = \sum_{j=0}^{T} \beta_{i,j} (V_j^l \mathbf{h}_j^l); \mathbf{h}_i^{l+1} = f(\mathbf{h}_i^l + g(\mathbf{h}_i^l)) \quad (5)$$

where Q^l, K^l, V^l are trainable parameters for layer l, d_k is the dimension of K, and the first layer of the hidden vector $\mathbf{h}_i^0$ refers to the record vector $\mathbf{r}_i$. To represent the full table, we apply max-pooling in Eq.3 using the last layer of hidden vectors similarly.

Method	E2E	WIKIBIO
Template	-	19.8
StructAware	-	44.89
Slug2Slug	66.19	-
MaxEnc	66.05	43.19
RnnEnc	66.11	44.93
SelfAtt	66.29	45.02

Table 3: Experimental results of different encoding methods and other systems over three datasets.

3 Experiments

3.1 Datasets and Evaluation Metrics

We conduct experiments on two datasets. E2E (Novikova et al., 2017) is a dataset for task-oriented language generation in the restaurant domain with 50,000 samples, the validation and test data are multi-reference; WIKIBIO (Lebret et al., 2016) contains 728,321 articles from English Wikipedia. It uses the first sentence of each article as the description of the corresponding infobox. The detailed statics of two datasets are listed in Table 2.

For evaluation metrics, we use BLEU-4 (Papineni et al., 2002) to assess the generation quality automatically.

3.2 Implementation Details

We tune all hyper-parameters according to the performance on the separated validation set. The dimension of trainable embeddings and hidden units in LSTMs are all set to 600. For the multi-layer and multi-head architecture, 3 layers and 4 multi-attention heads are used. During training, we regularize all layers with a dropout 0.1. For optimization, we use Adam with learning rate 0.0002. The gradient is truncated by 1. All experiments use beam size of 5 in decoding. We use pytorch version of OpenNMT (Klein et al., 2017) for implementation.

3.3 Performance

The results of different input encoding methods along with other competing systems on the test sets of three datasets are shown in Table 3. We compare three types of encoders (i.e., Pooling based encoders refer to MaxEnc, Recurrent encoders refer to RnnEnc, and self-attention encoders refer SelfAtt) introduced in Section 2.2 with the following generation systems: (1)

Methods	Training	E2E		WIKIBIO	
		Original	Shuffle	Original	Shuffle
MaxEnc	Original	66.05	66.05	43.19	43.19
	Shuffle	66.17	66.17	43.21	43.21
RnnEnc	Original	66.11	42.74	44.93	28.56
	Shuffle	64.08	65.40	43.95	43.59
SelfAtt	Original	66.29	66.29	45.02	45.02
	Shuffle	66.29	66.29	44.47	44.47

Table 4: Experimental results of different encoding methods trained and tested on different input settings

Template is a method that replaces the words occurring in both the table and the training sentences with a special token reflecting its field. (2) StructAware (Liu et al., 2018) is a structure-aware encoder-decoder architecture which using a modified LSTM unit and a specific attention mechanism to incorporate the attribute information. (3) Slug2Slug is an ensemble neural method that re-ranks several neural outputs during inference.

From table 3, the results show that three encoders achieves comparable results on E2E dataset, as the input of E2E is relatively short and simple. For WIKIBIO, the simple max-pooling encoder MaxEnc performs worse than the bidirectional LSTM encoder RnnEnc. The proposed method SelfAtt which capture the dependencies between arbitrary records yields better results compared to MaxEnc, and achieves comparable results with respect to RnnEnc. The result suggests modeling the dependencies among input records can yield better performance when the input is long and complex. More importantly, recurrent encoders only capture sequential dependencies, which is sensitive to the order of input records. To investigate the robustness of different table encoders, we design random shuffling noise over input data. For example, the original order of the input is "Birth name; Genres; Occupation; Associated Acts", and the order after random shuffling can be "'Geners; Birth name; Associated Acts; Occupation'. Note that we do not change the order of content inside a field. We apply such random shuffling noise on both training and testing stages. For model training, there are two choices: training on original input data Original or data with input random shuffling noise Shuffle. For testing, the trained model can be applied to the original input data or the shuffled version.

From the Table 4, We observe that the performance of RnnEnc drops dramatically when the input ordering information during training is different from the testing (i.e., the model trained on the original drops more than 15 BLEU scores when testing on input with random shuffling noise). In a slight difficult setting compared to original input, where both the input of training and testing are randomly shuffled, the performance of RnnEnc also decreases (i.e., 0.71 and 1.34 BLEU score decrease on E2E and WIKIBIO respectively). On the contrary, both MaxEnc and SelfAtt are less sensitive to the change of record ordering information, and the performance of both models under different shuffling noise are more stable than RnnEnc. The experimental results further confirms that using order invariant encoding SelfAtt is stable and suitable for table encoder.

	MaxEnc	RnnEnc	SelfAtt	ReSAtt
ESPN	14.01	16.32	13.93	16.43

Table 5: Results of different encoding methods over ESPN dataset

3.4 Limitations of Self-Attention Mechanism

The self-attention table encoder achieves comparable performance with respect to recurrent table encoders on E2E and WIKIBIO datasets. However, the input of these two datasets are relatively short. To investigate the performance of self-attention mechanism on capturing long range dependencies, we conduct SelfAtt on a recently proposed NBA dataset ESPN (Nie et al., 2018), where the average input length 165.9 and average field number is 134.2. The results on ESPN are shown in Table 5. The SelfAtt has difficulty in learning such long range dependencies and performs worse than RnnEnc. When applying a restricted self-attention ReSAtt (Wang

et al., 2018), where we limit the self-attention mechanism to capture dependencies within a fixed window size (set to 10 in the experiments), the result of `ReSAtt` performs comparable with respect to `RnnEnc`, despite this type of method is not order invariant. Handling long range dependencies for input data that is non-sensitive to the order of input data is a potential future work.

4 Conclusion

In this paper, we analyze several existing encoding methods for neural data-to-text generation. We find that modeling the dependency among the input records can yield better generation results. However, current recurrent table encoders can only model the sequential dependencies which is sensitive to the input order. We propose using self-attention for table encoder that can capture the dependencies and remains stable at the same time. In the future, we will analyze the explicit dependencies that lies in the input data, and improve the performance of encoding methods.

5 Acknowledgement

We thank the anonymous reviewers for helpful comments. The contact author of this paper, according to the meaning given to this role by Sun Yat-Sen University, is Rong Pan.

References

Gabor Angeli, Percy Liang, and Dan Klein. 2010. A simple domain-independent probabilistic approach to generation. In *EMNLP*, pages 502–512.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR*.

Regina Barzilay and Mirella Lapata. 2005. Collective content selection for concept-to-text generation. In *EMNLP*, pages 331–338.

Andrew Chisholm, Will Radford, and Ben Hachey. 2017. Learning to generate one-sentence biographies from wikidata. *CoRR*, abs/1702.06235.

Ondřej Dušek and Filip Jurcicek. 2016. Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. In *ACL*.

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.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *EMNLP*, pages 329–339.

Joohyun Kim and Raymond J. Mooney. 2010. Generative alignment and semantic parsing for learning from ambiguous supervision. In *COLING*, pages 543–551.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proc. ACL*.

Karen Kukich. 1983. Design of a knowledge-based report generator. In *ACL*, pages 145–150.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *EMNLP*, pages 1203–1213.

Percy Liang, Michael I. Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *ACL*, pages 91–99.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2018. Table-to-text generation by structure-aware seq2seq learning. In *AAAI*, pages 4881–4888. AAAI Press.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *NAACL*, pages 720–730.

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 *EMNLP*, pages 3879–3889. Association for Computational Linguistics.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, Saarbrücken, Germany. ArXiv:1706.09254.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

Laura Perez-Beltrachini and Claire Gardent. 2017. Analysing data-to-text generation benchmarks. In *INLG*, pages 238–242.

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

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *ACL*, pages 1073–1083. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 189–198. Association for Computational Linguistics.

Yizhong Wang, Sujian Li, and Jingfeng Yang. 2018. Toward fast and accurate neural discourse segmentation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 962–967. Association for Computational Linguistics.

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2017. Reference-aware language models. In *EMNLP*, pages 1850–1859.