RETRIEVAL ENHANCED SEGMENT GENERATION NEURAL NETWORK FOR TASK-ORIENTED DIALOGUE SYSTEMS

Miaoxin Chen [1], *Zibo Lin* [2], *Rongyi Sun* [1], *Kai Ouyang* [1], *Hai-Tao Zheng* [1⋆], *Rui Xie* [3], *Wei Wu* [3]

[1] Shenzhen International Graduate School, Tsinghua University, China
[2] WeChat Search Application Department, Tencent, China
[3] Meituan, Shanghai, China

ABSTRACT

For task-oriented dialogue systems, Natural Language Generation (NLG) is the last and vital step which aims at generating an appropriate response according to the dialogue act (DA). While end-to-end neural networks have achieved promising performances on this task, the existing models still struggle to avoid slot mistakes. To address this challenge, we propose a novel segmented generation approach in this paper. The proposed method operates by progressively generating text for the span between two adjacent keywords (act type and slots) in semantically ordered DA. This procedure is recursively applied from left to right until a response is completed. Besides, a retrieval mechanism is utilized to better match the diversity and fluency in human language. Experimental results on four datasets demonstrate that our model achieves state-of-the-art slot error rate and also gets competitive performance on BLEU score with all strong baselines.

Index Terms— Task-oriented dialogue system, NLG

1. INTRODUCTION

Natural Language Generation (NLG), as the final and crucial component of task-oriented dialogue systems, aims at converting a meaning representation, i.e., dialogue act (DA), into a natural language response. Traditional approaches [1, 2, 3] are mostly rule-based. Despite their robustness, they heavily rely on handcrafted rules and domain-specific knowledge. Recently, data-driven approaches [4, 5, 6, 7, 8] have achieved promising performances on this task. Specifically, Zhu et al.[6] and Li et al.[7] have designed sophisticated networks to generate fluent and explicable responses.

However, those existing models still struggle to avoid slot mistakes, e.g., neglecting an input slot and generating a redundant slot. As shown in Table 1, following the act type *inform*, there are two slot-value pairs which are expected to appear rightly in the response. Compared to the reference, mistaken generation includes the missing error of slot [NAME] and the redundant error of slot [NEAR].

Input DA	inform (PRICERANGE = cheap, NAME = Ploy 2)
Reference	I have found Ploy 2 , which is in a cheap price range
Missing	It is a cheap restaurant [NAME]
Redundant	I have found Ploy 2 , with a cheap price range near the park [NEAR]

Table 1. Example of task-oriented dialogue and its mistaken generation (missing slot in red and redundant slot in blue).

To address this challenge, we propose a novel Retrieval Enhanced Segment Generation Neural Network (RESGNN). Our model consists of two Transformer [9] based modules, e.g., a sentence planner and a segment generator. According to the observation that the order of values in response is not restricted by the input DA, we first apply the sentence planner to output a skeleton including an act type token and several semantically ordered slots. The sentence planner can guarantee that all input slots will be reserved in the output. Then the segment generator progressively generates text for the span between two adjacent keywords[1] in the skeleton. Inspired by previous works[10, 11], we adopt the delexicalization technique to simplify the input, which helps our model make fewer numerical mistakes. To eliminate the semantic discontinuity and awkward expression of our two-step model, we further introduce a retrieval mechanism to improve the diversity and fluency of generated responses.

The contributions of our paper are threefold: a) We propose a novel segmented generation approach to tackle the problem of slot mistakes. b) A retrieval mechanism is applied to make the model better mimic the human language. c) Experiments on four datasets demonstrate that our model

⋆ Corresponding author: zheng.haitao@sz.tsinghua.edu.cn. This research was supported by Meituan, National Natural Science Foundation of China (Grant No.6201101015), Beijing Academy of Artificial Intelligence (BAAI), Natural Science Foundation of Guangdong Province (Grant No.2021A1515012640), the Basic Research Fund of Shenzhen City (Grant No.JCYJ20210324120012033 and JCYJ20190813165003837), and Overseas Cooperation Research Fund of Tsinghua Shenzhen International Graduate School (Grant No.HW2021008).

[1]We define the combination of act type and slot as *keywords*

Stage	Text sequence
X^0	<inform> [NAME] [PHONE] [SEP]
X^1	<inform> I have found [NAME] [PHONE] [SEP]
X^2	<inform> I have found [NAME] and [PHONE] [SEP]
X^3	<inform> I have found [NAME] and [PHONE] is the phone number [SEP]

Table 2. Example of the progressive generation process from the RESGNN model. Words in blue are newly generated at the current stage. [SEP] is a special slot added to take care of the text following the last slot in skeleton.

achieves state-of-the-art slot error rate (ERR) and gets competitive performance on BLEU score with all strong baselines.

2. RELATED WORK

We propose our model with inspiration from both task-oriented NLG and hard-constrained NLG. For task-oriented dialogues, conventional methods [2, 3] are mostly rule-based and tend to generate rigid responses. Recently, data-driven approaches with end-to-end neural networks [4, 5, 7, 8] have attached more attention. The pioneer [4] designs an encoder-decoder model with an attention mechanism to take account of input slot-value pairs. Li et al.[7] propose an iterative rectification network to tackle the challenge of slot mistakes and achieve state-of-the-art results according to ERR. However, it still has no powerful means to avoid slot mistakes. Once the iterations limit is reached, the rectification has to stop.

The hard-constrained text generation task aims to generate a complete text according to a list of keywords, where the keywords have to be exactly included in the final generated text with the same order. Zhang et al.[12] design an insertion-based model [13] which works by progressively inserting new tokens between existing tokens in a parallel manner. Because of its non-autoregressive character, the quality of the output sentences still needs to be further improved.

3. METHOD

3.1. Overview

Our method decomposes the whole generation process into two steps: plan, then generate. We first use the delexicalization technique to replace the values with their slot names for all data during training and inference. Then define $X^0 = \{x_0, x_1, ..., x_T\}$ as the skeleton generated by sentence planner based on the input DA, where x_0 is the act type token and each of $x_1, ..., x_T$ is a token representing a particular slot. We treat the following process as a hard-constrained text generation task. As shown in Table 2, the generation procedure is formulated as a progressive sequence of K stages:

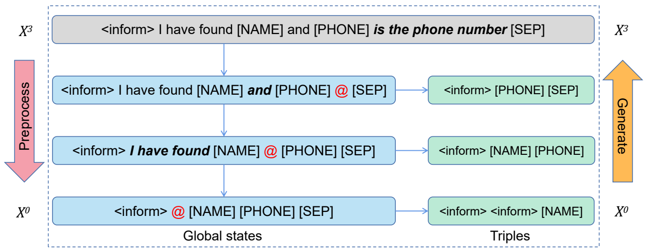
X³
<inform> I have found [NAME] and [PHONE] is the phone number [SEP]
X³
Preprocess
<inform> I have found [NAME] and [PHONE] @ [SEP]
<inform> [PHONE] [SEP]
Generate
<inform> I have found [NAME] @ [PHONE] [SEP]
<inform> [NAME] [PHONE]
X⁰
<inform> @ [NAME] [PHONE] [SEP]
<inform> <inform> [NAME]
X⁰
Global states
Triples

Fig. 1. Example of the data preparation. Blocks in blue represent the global states and blocks in green represent the triples which is combined by the act type and keywords on both sides of token @. For $k \in \{1, ..., K\}$, the bold text in X^k is the label or generated text of stage X^{k-1}. The generation process reverses the above preparation process.

$S = \{X^0, X^1, ..., X^K\}$. At last, we replace all slots in the response template, e.g., X^K, with their corresponding values to get the final response.

3.2. Data preparation

Difference from previous models which directly generate the responses, we prepare data in a two-step manner that eases our model training. As illustrated in Fig.1, for a delexicalized response, we replace the text between two neighbor keywords with a special token @ from right to left. By this way we can get a *global state* and corresponding *triple* as the input of segment generator for each stage. Token @ is used to explicitly indicate the location where the text should be generated. For cases where no words need to be generated between two keywords, we use a special [nothing] token as the label.

For the retrieval process, we take triple as the key and collect all the subtexts with the same key to obtain a particular set. Note that most sets have a variety of subtexts because of the diversification of human language and the different sentence structures of responses.

3.3. Sentence planner

We implement the sentence planner by a vanilla Transformer [9] with constrained decoding. In specific, the sentence planner takes the delexicalized DA as input, which is a list of act type token and some slots in alphabetical order. The output is another list consists of semantically ordered slots. For the decoder of the Transformer, we use the act type as its first input token. Furthermore, to ensure that the input and output slots differ only sequentially, in the inference processing, we restrict the model to selecting tokens only from the slots in input step by step, and selected slots cannot be selected again until all slots in the input have been generated. Then we can get the skeleton X^0 by concatenating the act type token and generated slots.

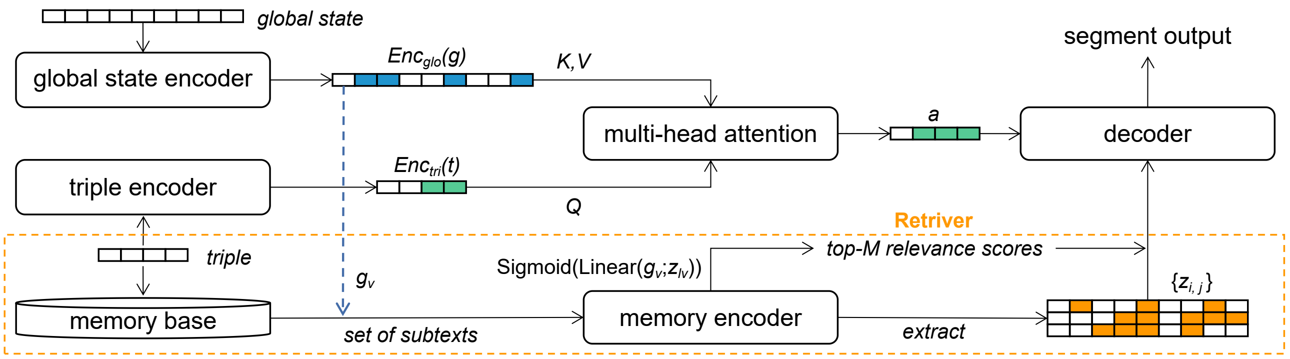

Fig. 2. Illustration of the segment generator. It generates a piece of text for each stage to construct the response template.

3.4. Segment generator

The segment generator is responsible for generating the response template recursively based on the skeleton. As shown in Fig.2, for each stage in $\{X^0, X^1, ..., X^{K-1}\}$, global state encoder and triple encoder (denote Enc_{glo} and Enc_{tri}) first encode the global state and the triple respectively, both of which are prepended the [BOS] token to their token sequence. We then introduce a multi-head attention layer to fuse the representations from the two encoders above and pass the output a to the decoder. The procedure can be written as:

$$a = \text{MultiHead}\left(\text{Enc}_{tri}(t), \text{Enc}_{glo}(g), \text{Enc}_{glo}(g)\right) \quad (1)$$

where t and g represent triple and global state respectively.

The triple is also used to get the corresponding set of subtexts from the memory base for the retrieval process. In the retriever, a memory encoder Enc_{mem} is utilized to map the l th subtext z_l of the set to its hidden representation. Then we can get the relevance score s_l between the l th subtext and the global state of the current stage as follows:

$$s_l = \text{Sigmoid}(\text{Linear}\left(g_v; z_{lv}\right)) \quad (2)$$

where ; means concatenation, g_v and z_{lv} are the hidden representations at [BOS] token in $\text{Enc}_{glo}(g)$ and $\text{Enc}_{mem}(z_l)$.

Our decoder generates a subtext y in an auto-regressive manner. At each time step t, the decoder takes both a and $y_{1:t-1}$ to generate a hidden state h_t which is then converted to next-token probabilities P_v through a linear projection followed by softmax function. To accommodate the extra memory input, we first retrieve top-M subtexts according to the relevance score. Then we simply extract a contextualized token embeddings set $\{z_{i,k}\}_{k=1}^{L_i}$ from $\{\text{Enc}_{mem}(z_i)\}$, where z_i is a subtext in the retrieved ones and L_i is the length of z_i. Inspired by [14], to enable the gradient flow from the decoder to the network which calculates the relevance scores in Eq.2, we send s_i to the decoder as the bias of z_i. By this way, a cross attention over all retrieved subtexts can be obtained by:

$$\alpha_{ij} = \frac{\exp\left(h_t^T W_m z_{i,j} + \beta s_i\right)}{\sum_{i=1}^{M}\sum_{k=1}^{L_i}\exp\left(h_t^T W_m z_{i,k} + \beta s_i\right)} \quad (3)$$

$$c_t = W_c \sum_{i=1}^{M}\sum_{j=1}^{L_i} \alpha_{ij} z_{i,j} \quad (4)$$

in which α_{ij} is the attention score of the j th token in z_i, c_t is a weighted combination of retrieved token embeddings, s_i is the relevance score of z_i, β is a trainable scalar, W_m and W_c are both trainable matrices. Then h_t is updated by the sum of h_t and c_t. Besides, the attention score is used as a probability of copying its corresponding token [15, 16]. Finally, the next-token probabilities are computed as:

$$p\left(y_t\right) = \lambda_t P_v\left(y_t\right) + \left(1 - \lambda_t\right) \sum_{i=1}^{M}\sum_{j=1}^{L_i} \alpha_{ij} \mathbf{I}_{z_{ij}=y_t} \quad (5)$$

where $\mathbf{I}$ is the indicator function and λ_t is a gating probability computed by a feed-forward network as $ff(h_t, c_t)$.

4. EXPERIMENTS

4.1. Experimental Setup

Datasets. We use four datasets to evaluate our model. The Hotel and Restaurant datasets are proposed in [11], and Laptop and TV datasets are released by [17]. Datasets above are with 5.2K, 5.4K, 13.2K and 7.0K samples respectively. We follow the automatic evaluation metrics used in [11, 17], e.g., BLEU and ERR which is calculated as $(p + q)/N$ where N is the total number of slots in the DA, and p, q is the number of missing and redundant slots in the generated template.

Details. We optimize the parameters of sentence planner and segment generator independently in training. For the segment generator, all the data from different responses and in different stages with ground-truth slots order are scrambled and trained together. For the triples not in memory base, we retrieve the subtext-sets of their similar triples . To enhance the robustness of the model, we further finetune the saved model if no new one is saved after 10 epochs. Specifically, the global states of randomly half training data will be replaced by the ones generated by the model itself.

Settings. We set 8 attention heads, 256 dimensional hidden

Model	Restaurant		Hotel		Laptop		Television	
	BLEU	ERR(%)	BLEU	ERR(%)	BLEU	ERR(%)	BLEU	ERR(%)
HLSTM [11]	0.747	0.74	0.850	2.67	0.513	1.10	0.525	2.50
SCLSTM [10]	0.753	0.38	0.848	3.07	0.512	0.79	0.527	2.31
RALSTM [5]	0.779	0.16	0.898	0.43	0.525	0.42	0.541	0.63
NLG-LM [6]	0.795	-	**0.939**	-	0.586	-	**0.617**	-
IRN(+KNN) [7]	0.807	0.11	0.911	0.32	0.537	0.29	0.559	0.35
Interpretable NLG [8]	0.812	0.20	0.923	0.46	**0.591**	0.31	0.610	0.51
RESGNN (ours)	**0.817**	**0.00**	0.913	**0.00**	0.572	**0.00**	0.597	**0.04**

Table 3. Experiment results on four datasets for all baselines and our model. We follow the baseline results reported in [7, 8].

Model	BLEU	Human Evaluation		
		Rel.	Flu.	Div.
RESGNN	**0.572 ($\pm$0.002)**	**4.32**	**3.85**	**3.44**
- finetune	0.567 ($\pm$0.002)	4.31	3.72	3.42
- retriver	0.564 ($\pm$0.003)	4.20	3.54	3.38

Table 4. Ablation study and human evaluation on Laptop. For BLEU score, the confidence is set to 0.95 under t-test.

state, and 1024 dimensional feed-forward state to all Transformer blocks in our model. All encoders and decoders in our model are four-layer and six-layer Transformer blocks respectively. The dropout rate is 0.4 and 0.3 for sentence planner and segment generator respectively. We retrieve the top-5 subtexts for each triple. We set learning rate as 0.001 and use greedy search for decoding of the segment generator.

4.2. Main Results

We choose 6 baselines for comparison: HLSTM [11] designs a heuristic gate for the states of slots and values. SCLSTM [10] adds an extra reading gate to improve standard LSTM. RALSTM [5] proposes a RNN-based decoder which efficiently utilizes the attention mechanism. NLG-LM [6] greatly improves BLEU scores through a multi-task framework. IRN [7] presents an iterative rectification network to reduce the slot mistakes. Interpretable NLG [8] generates the responses in a more explicable way compared to previous models.

As shown in Table 3, our model achieves an absolute advantage in ERR across four datasets. On one hand, our model generates subtexts for each span recursively, so it will never miss a slot in input. On the other hand, the slot tokens never appear in the labels during the training process, so the probability of generating a redundant slot is extremely low for our model. We get exact zero ERR on three datasets and a 0.04% ERR on TV dataset because of the mislabeling. For BLEU scores, our model is also competitive with current state-of-the-art methods and gains remarkable performance improvement on the Restaurant dataset. The reason we can not get the best performance on other three datasets in BLEU is probably because our two-steps method suffers from the accumulative error and the special token [nothing] will greatly affect the subsequent generation process if it is chosen wrongly.

Input DA: recommend (NAME=crios 93, AUDIO=nicam stereo, SIZE=large, FAMILY=l1)

Reference: the [SIZE] [NAME] television in the [FAMILY] **family** features [AUDIO] audio

HLSTM Generation: the [NAME] is in [FAMILY] with [SIZE] screen and cost about [PRICE] (with missing error on [AUDIO] and redundant error on [PRICE])

Our Generation: the [NAME] television is in the [FAMILY] **product family** with a [SIZE] screen size and [AUDIO] audio equipment

Fig. 3. One case from TV, same slots are in the same color.

4.3. Ablation Study and Human Evaluation

We conduct ablation study together with human evaluation. Human evaluators are asked to score each response according to: 1) Relevance (whether a response is closely related to the DA, noted as Rel.), 2) Fluency (noted as Flu.), 3) Diversity (noted as Div.). The score of each aspect ranges from 1 to 5 and the higher score means better. Results in Table 4 demonstrate that the retrieval mechanism and the finetune process do improve the diversity and fluency of the responses.

4.4. Case Study

One case is shown in Fig.3, For this example, the HLSTM model failed to generate ***family*** or ***product family*** to follow the [FAMILY] token which will be replaced by value *l1*, making the response be with less naturalness compared to the reference and our generation. Although the order of the slots is different from the reference, the output of our model is still a well-organized response template without any slot mistake.

5. CONCLUSION

In this paper, we propose RESGNN for task-oriented dialogue systems. Experimental results on four datasets show that our method has great robustness to avoid slot mistakes and also gets a promising performance to match the human language.

6. REFERENCES

[1] Amanda Stent, Rashmi Prasad, and Marilyn Walker, "Trainable sentence planning for complex information presentations in spoken dialog systems," in *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, 2004, pp. 79–86.

[2] Yuk Wah Wong and Raymond Mooney, "Generation by inverting a semantic parser that uses statistical machine translation," in *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, 2007, pp. 172–179.

[3] Ioannis Konstas and Mirella Lapata, "A global model for concept-to-text generation," *Journal of Artificial Intelligence Research*, vol. 48, pp. 305–346, 2013.

[4] Ondřej Dušek and Filip Jurcicek, "Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2016, pp. 45–51.

[5] Van-Khanh Tran and Minh Le Nguyen, "Natural language generation for spoken dialogue system using rnn encoder-decoder networks," in *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, 2017, pp. 442–451.

[6] Chenguang Zhu, Michael Zeng, and Xuedong Huang, "Multi-task learning for natural language generation in task-oriented dialogue," 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)*, 2019, pp. 1261–1266.

[7] Yangming Li, Kaisheng Yao, Libo Qin, Wanxiang Che, Xiaolong Li, and Ting Liu, "Slot-consistent nlg for task-oriented dialogue systems with iterative rectification network," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 97–106.

[8] Yangming Li and Kaisheng Yao, "Interpretable nlg for task-oriented dialogue systems with heterogeneous rendering machines," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021, vol. 35, pp. 13306–13314.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[10] Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young, "Semantically conditioned lstm-based natural language generation for spoken dialogue systems," *arXiv preprint arXiv:1508.01745*, 2015.

[11] Tsung-Hsien Wen, Milica Gasic, Dongho Kim, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young, "Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking," in *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, 2015, pp. 275–284.

[12] Yizhe Zhang, Guoyin Wang, Chunyuan Li, Zhe Gan, Chris Brockett, and William B Dolan, "Pointer: Constrained text generation via insertion-based generative pre-training," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 8649–8670.

[13] Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit, "Insertion transformer: Flexible sequence generation via insertion operations," in *Proceedings of the 36th International Conference on Machine Learning*, Kamalika Chaudhuri and Ruslan Salakhutdinov, Eds. 09–15 Jun 2019, vol. 97 of *Proceedings of Machine Learning Research*, pp. 5976–5985, PMLR.

[14] Deng Cai, Yan Wang, Huayang Li, Wai Lam, and Lemao Liu, "Neural machine translation with monolingual translation memory," 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)*, 2021, pp. 7307–7318.

[15] Abigail See, Peter J Liu, and Christopher D Manning, "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)*, 2017, pp. 1073–1083.

[16] Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung, "Transferable multi-domain state generator for task-oriented dialogue systems," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 808–819.

[17] Tsung-Hsien Wen, Milica Gašic, Nikola Mrkšic, Lina M Rojas-Barahona, Pei-Hao Su, David Vandyke, and Steve Young, "Multi-domain neural network language generation for spoken dialogue systems," in *Proceedings of NAACL-HLT*, 2016, pp. 120–129.