Semantic Refinement GRU-Based Neural Language Generation for Spoken Dialogue Systems

Van-Khanh Tran[1,2](✉) and Le-Minh Nguyen[1]

[1] Japan Advanced Institute of Science and Technology, JAIST, 1-1 Asahidai, Nomi, Ishikawa 923-1292, Japan
{tvkhanh,nguyenml}@jaist.ac.jp

[2] University of Information and Communication Technology, ICTU, Thai Nguyen University, Thai Nguyen, Vietnam
tvkhanh@ictu.edu.vn

Abstract. Natural language generation (NLG) plays a critical role in spoken dialogue systems. This paper presents a new approach to NLG by using recurrent neural networks (RNN), in which a gating mechanism is applied before RNN computation. This allows the proposed model to generate appropriate sentences. The RNN-based generator can be learned from unaligned data by jointly training sentence planning and surface realization to produce natural language responses. The model was extensively evaluated on four different NLG domains. The results show that the proposed generator achieved better performance on all the NLG domains compared to previous generators.

1 Introduction

Natural Language Generation is a critical component in a Spoken Dialogue System (SDS), and its task is to convert a meaning representation produced by the dialogue manager into natural language utterances. Conventional approaches to NLG follow a pipeline which typically breaks down the task into sentence planning and surface realization. Sentence planning decides the order and structure of sentence, which is followed by a surface realization which converts the sentence structure into the final utterance. Previous approaches to NLG still rely on extensive hand-tuning templates and rules that require expert knowledge of linguistic representation. There are some common and widely approaches to solve NLG problems, including rule-based [3], corpus-based n-gram models [10], and a trainable generator [15]. Joint based generators use a two-step pipeline [4,14]; or applying a joint model for both tasks [19,22].

Recently, approaches based on recurrent neural networks have shown advantages in solving the NLG tasks. RNN-based models have been used for NLG as a joint training model [19,22] and an end-to-end training model [23]. A recurring problem in such systems is requiring datasets annotated for specific dialogue acts[1] (DA). Moreover, previous works may lack ability to handle cases such as the binary slots (*i.e.*, *yes* and *no*) and slots that take *don't_care* value which cannot be directly delexicalized [19], or to generalize to unseen domains [22]. Furthermore, a problem of the current generators is that the generators produce the next token based on the information from the forward context, whereas the sentence may depend on the backward context. As a result, the generators tend to generate nonsensical utterances.

We propose a statistical NLG based on a gating mechanism on a GRU model, in which the gating mechanism is applied before RNN computation. The proposed model can learn from unaligned data by jointly training the sentence planning and surface realization to generate required sentences. We found that the proposed model can produce sentences in a more correct oder than the existing models. The previous RNN-based generators may have lack of consideration about the order of slot-value pairs during generation. For example, given a DA with pattern: *Compare*(name = A, property1 = *a1*, property2 = *a2*, name = B, property1 = *b1*, property2 = *b2*). The pattern for correct utterances can be: [*A-a1-a2, B-b1-b2*], [*A-a2-a1, B-b2-b1*], [*B-b1-b2, A-a1-a2*], [*B-b1-b2, A-a2-a1*]. Therefore, a generated utterance: "The A has *a1* and ***b1*** properties, while the B has ***a2*** and *b2* properties" is an incorrect utterance, in which ***b1*** and ***a2*** properties were generated in wrong order. This occasionally leads to inadequate sentences.

We assessed the proposed generators on varied NLG domains, in which the results showed that our proposed method outperforms the previous methods in terms of BLEU [11] and slot error rate ERR [22] scores. To summary, we make three contributions in this study where we: (i) propose two semantic refinement RNN-based models, in which a gating mechanism is applied before computational RNN to refine the original inputs, (ii) conduct extensively experiments on four NLG domains, and (iii) analyze the effectiveness of the proposed models on ability to handle the undelexicalized tokens, and to generalize to unseen domain when limited amount of in-domain data was fed.

2 Related Work

Conventional approaches to NLG traditionally split the task into two subtasks: sentence planning and surface realization. Sentence planning deals with mapping of the input semantic symbols onto a linguistic structure, *e.g.*, a tree-like or a template structure. The surface realization then converts the structure into an appropriate sentence [15]. Despite their success and wide use in solving NLG problems, these traditional methods still rely on the handcrafted rule-based generators or rerankers. The authors in [10] proposed a class-based n-gram language model (LM) generator which can learn to generate the sentences for a given DA and then select the best sentences using a rule-based reranker. Some of the limitation of the class-based LMs were addressed in [13] by proposing a method

[1] A combination of an action type and a list of slot-value pairs. *e.g. inform(name = 'Frances'; area = 'City Center').*

based on a syntactic dependency tree. A phrase-based generator based on factored LMs was introduced in [8], which can learn from a semantically aligned corpus.

Recently, RNNs-based approaches have shown promising performance in the NLG domain. The authors in [6,17] used RNNs in a multi-modal setting to generate captions for images, while a generator using RNNs to create Chinese poetry was also proposed in [24]. The authors in [7] encoded an unstructured textual knowledge source along with previous responses and context to produce a response for technical support queries. For task-oriented dialogue systems, a combination of a forward RNN generator, a CNN reranker, and a backward RNN reranker was proposed in [19] to generate utterances. A semantically conditioned-based Long Short-Term Memory (LSTM) generator was introduced in [22], which proposed a control "*reading*" gate to the traditional LSTM cell and can learn the gating mechanism and language model jointly. A recurring problem in such systems is the lack of sufficient domain-specific annotated data.

3 Recurrent Neural Language Generator

The recurrent language generator proposed in this paper based on a RNN language model [9], which consists of three layers: an input layer, a hidden layer and an output layer. The input to the network at each time step t is a 1-hot encoding $\mathbf{w}_t$ of a token[2] w_t which is conditioned on a recurrent hidden layer $\mathbf{h}_t$. The output layer $\mathbf{y}_t$ represents the probability distribution of the next token given previous token w_t and hidden $\mathbf{h}_t$. We can sample from this conditional distribution to obtain the next token in a generated string, and feed it as the next input to the generator. This process finishes when a stop sign is generated [6], or some constraint are reached [24]. The network can generate a sequence of tokens which can be lexicalized[3] to form the required utterance. Moreover, in order to ensure that the generated utterance represents the intended meaning of the given DA, the generator is further conditioned on a vector $\mathbf{z}$, a 1-hot vector representation of DA. Inspired by work in [18], we propose an intuition: *Gating before computation*, in which we add gating mechanism before the RNN computation to semantically refine the input tokens. The following sections present two proposed Semantic Refinement (SR) gating based RNN generators.

3.1 SRGRU-BASE

In this model, instead of feeding an input token $\mathbf{w}_t$ to the RNN model at each time step t, the input token is filtered by a semantic gate which is computed as follows:

$$\begin{aligned} \mathbf{d}_t &= \sigma(\mathbf{W}_{dz}\mathbf{z}) \\ \mathbf{x}_t &= \mathbf{d}_t \odot \mathbf{w}_t \end{aligned} \tag{1}$$

[2] Input texts are delexicalized in which slot values are replaced by its corresponding slot tokens.

[3] The process in which slot token is replaced by its value.

where: $\mathbf{W}_{dz}$ is a trained matrix to project the given DA representation into the word embedding space, and $\mathbf{x}_t$ is new input. Here $\mathbf{W}_{dz}$ plays a role of sentence planning since it can directly capture which DA features are useful during the generation to encode the input information. The $\odot$ element-wise multiplication plays a part in word-level matching which learns not only the vector similarity, but also preserve information about the two vectors. $\mathbf{d}_t$ is called a *refinement* gate since the input tokens are refined by the DA information. As a result, we can represent the whole input sentence based on these refined inputs using RNN model.

In this study, we use GRU, which was recently proposed in [2], instead of LSTM as building computational block for RNN, which is formulated as follows:

$$\mathbf{r}_t = \sigma(\mathbf{W}_{rx}\mathbf{x}_t + \mathbf{W}_{rh}\mathbf{h}_{t-1}) \tag{2}$$

$$\mathbf{u}_t = \sigma(\mathbf{W}_{ux}\mathbf{x}_t + \mathbf{W}_{uh}\mathbf{h}_{t-1}) \tag{3}$$

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_{hx}\mathbf{x}_t + r_t \odot \mathbf{W}_{hh}\mathbf{h}_{t-1}) \tag{4}$$

$$\mathbf{h}_t = \mathbf{u}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{u}_t) \odot \tilde{\mathbf{h}}_t \tag{5}$$

where: $\mathbf{W}_{rx}, \mathbf{W}_{rh}, \mathbf{W}_{ux}, \mathbf{W}_{uh}, \mathbf{W}_{hx}, \mathbf{W}_{hh}$ are weight matrices; $\mathbf{r}_t$, $\mathbf{u}_t$ are reset and update gate, respectively, and $\odot$ denotes for element-wise product. The Semantic Refinement GRU (SRGRU)-Base architecture is shown in Fig. 1.

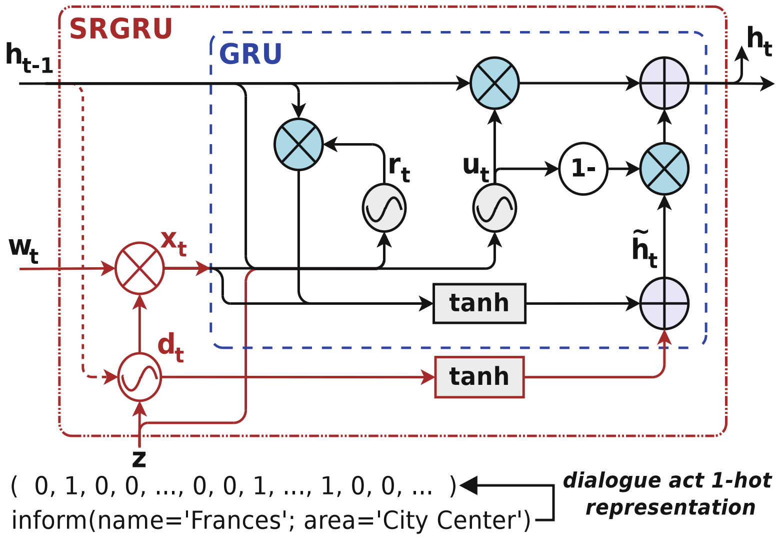


Fig. 1. SRGRU-Context cell. The blue dashed box is a traditional GRU cell in charge of surface realization, while the red parts form sentence planning based on a sigmoid control gate $\mathbf{d}_t$ and a dialogue act $\mathbf{z}$. The contextual information $\mathbf{h}_{t-1}$ is imported into the refinement gate $\mathbf{d}_t$ via red dotted line. The SRGRU-Base is achieved by omitting this link. (Color figure online)

Finally, the output distribution of each token is defined by applying a softmax function g as follows:

$$P(w_{t+1} \mid w_t, w_{t-1}, ...w_0, \mathbf{z}) = g(\mathbf{W}_{ho}\mathbf{h}_t) \tag{6}$$

where: $\mathbf{W}_{ho}$ is learned linear projection matrix. At training time, we use the ground truth token for the previous time step in place of the predicted output. At test time, we implement a simple beam search to over-generate several candidate responses.

3.2 SRGRU-CONTEXT

SRGRU-Base uses only the DA information to gate the input sequence token by token. As a results, this gating mechanism may not capture the relationship between multiple words. In order to import context information into the gating mechanism, the Eq. 1 is modified as follows:

$$\begin{aligned} \mathbf{d}_t &= \sigma(\mathbf{W}_{dz}\mathbf{z} + \mathbf{W}_{dh}\mathbf{h}_{t-1}) \\ \mathbf{x}_t &= \mathbf{d}_t \odot \mathbf{w}_t \end{aligned} \tag{7}$$

where: $\mathbf{W}_{dz}$ and $\mathbf{W}_{dh}$ are weight matrices. $\mathbf{W}_{dh}$ acts like a key phrase detector that learns to capture the pattern of generation tokens or the relationship between multiple tokens. In other words, the new input $\mathbf{x}_t$ consists of information of the original input token $\mathbf{w}_t$, the dialogue act $\mathbf{z}$, and the hidden context $\mathbf{h}_{t-1}$. $\mathbf{d}_t$ is called the *refinement* gate because the input tokens are refined by a combination gating information of the dialogue act $\mathbf{z}$ and the previous hidden state $\mathbf{h}_{t-1}$. By taking advantage of gating mechanism from the LSTM model [5] in which the gating mechanism is employed to solve the gradient vanishing and exploding problem, we propose to apply the refinement gate deeper into the GRU cell. Firstly, the GRU reset and update gates can be further influenced on the given dialogue act $\mathbf{z}$ and the refined input $\mathbf{x}_t$. The Eqs. (2) and (3) are modified as follows:

$$\mathbf{r}_t = \sigma(\mathbf{W}_{rx}\mathbf{x}_t + \mathbf{W}_{rh}\mathbf{h}_{t-1} + \mathbf{W}_{rz}\mathbf{z}) \tag{8}$$

$$\mathbf{u}_t = \sigma(\mathbf{W}_{ux}\mathbf{x}_t + \mathbf{W}_{uh}\mathbf{h}_{t-1} + \mathbf{W}_{uz}\mathbf{z}) \tag{9}$$

where: $\mathbf{W}_{rz}$ and $\mathbf{W}_{uz}$ act like background detectors that learn to control the style of the generating sentence. Secondly, Eq. (4) is modified so that the candidate activation $\tilde{\mathbf{h}}_t$ also depends on the refinement gate,

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_{hx}\mathbf{x}_t + \mathbf{r}_t \odot \mathbf{W}_{hh}\mathbf{h}_{t-1}) + \tanh(\mathbf{W}_{dc}\mathbf{d}_t) \tag{10}$$

By this way, the reset and update gates learn not only the long-term dependency but also the gating information from the dialogue act and the previous hidden state. We call the resulting architecture Semantic Refinement GRU (SRGRU)-Context which is shown in Fig. 1.

3.3 Training

The cost function was the negative log-likelihood and computed by:

$$F(\theta) = -\sum_{t=1}^{T} \mathbf{y}_t^\top \log \mathbf{p}_t \tag{11}$$

where $\mathbf{y}_t$ is the ground truth word distribution, $\mathbf{p}_t$ is the predicted word distribution, T is length of the input sequence. The generators were trained by treating each sentence as a mini-batch with the l_2 regularization was added to the cost function for every 10 training examples. The models were initialized with pre-trained word vectors GLOVE [12] and optimized by using stochastic gradient descent and back propagation through time. To prevent over-fitting, early stopping was implemented using a validation set.

3.4 Decoding

The decoding phase we employ here is similar to [16] which consists of over-generation and re-ranking phases. The forward generator, in the over-generation phase, is conditioned on the given DA uses a beam search algorithm to generate candidate utterances, whereas the cost of forward generator $F_{fw}(\theta)$, in the re-ranking phase, is computed to form the re-ranking score R as follows:

$$R = F_{fw}(\theta) + \lambda ERR \tag{12}$$

where λ is a trade off constant which is set to a large value in order to severely penalize nonsensical outputs. The slot error rate ERR, which is the number of generated slots that are either redundant or missing, is computed by:

$$ERR = \frac{p + q}{N} \tag{13}$$

where N is the total number of slots in DA, and p, q is the number of missing and redundant slots, respectively. The ERR re-ranking criteria as mentioned in [22] cannot handle arbitrary slot-value pairs, *i.e. binary* slots or slots that take *don't_care* value, because such these pairs cannot be delexicalized and matched.

4 Experiments

4.1 Datasets

We conducted experiments using four different NLG domains: finding a restaurant, finding a hotel, buying a laptop, and buying a television. The Restaurant and Hotel domains were collected in [22] which contain system dialogue acts, shared slots, and specific domain slots. The Laptop and TV datasets have been released in [20] with about 13K distinct DAs in the Laptop and 7K distinct DAs in the TV. These two datasets have a much larger input space but only one training example for each DA so that the system must learn partial realization of concepts and be able to recombine and apply them to unseen DAs. The number of dialogue act types and slots of datasets is also larger than in Restaurant and Hotel datasets. As a result, the NLG tasks for the Laptop and TV datasets become much harder.

4.2 Experimental Setups

The generators were implemented using the TensorFlow library [1] and trained by partitioning each of the datasets into training, validation and testing set in the ratio 3:1:1. The hidden layer size was set to be 80, and the generators were trained with a 70% of dropout rate. We perform 5 runs with different random initialization of the network and the training is terminated by using early stopping as described in Sect. 3.3. We select model that yields the highest BLEU score on the validation set. The decoder procedure used beam search with a beam width of 10. We set λ to 1000 to severely discourage the reranker from selecting utterances which contain either redundant or missing slots. For each DA, we over-generated 20 candidate utterances and selected the top 5 realizations after reranking. Because the proposed models work stochastically, except the results reported in Table 1, all the results shown were averaged over 5 randomly initialized networks.

Since some sentences may depend on both the past and the future during generation, we train another backward SRGRU-Context to utilizing the flexibility of the refinement gate $\mathbf{d}_t$, in which we tie its weight matrices such $\mathbf{W}_{dz}$ and $\mathbf{W}_{dh}$ (Eq. 7) for both. We found that by tying matrix $\mathbf{W}_{dz}$ for both forward and backward RNNs, the proposed generator seems to produce more correct and grammatical utterances than those having the only forward RNN. This model called Tying Backward SRGRU-Context (TB-SRGRU).

In order to better understand the effectiveness of the proposed model, we conduct more experiments to compare the SRGRU-Context with the previous generator SCLSTM in a variety of setups on proportion of training corpus, beam size, and top-k best results. Firstly, the Restaurant and TV datasets were chosen, in which the SCLSTM model obtained the best performances and the NLG task comes from a limited domain to a more diverse domain as described in Sect. 4.1. In this setup, the models were run with different size of training corpus. Secondly, we examined the stability of SRGRU-Context model on different setups of beam size and top-k best results.

4.3 Evaluation Metrics and Baselines

The generator performance was assessed by using two objective evaluation metrics, the BLEU score and the slot error rate ERR. Note that the slot error rate ERR was computed as an auxiliary metric alongside the BLEU score and calculated by averaging slot errors over each of the top 5 realizations in the entire corpus. Both metrics were computed by adopting code from an open source benchmark toolkit for Natural Language Generation[4].

We compared our proposed models against with the general GRU (GRU-Base) and three strong baselines released from the NLG toolkit:

- ENCDEC proposed in [21] which applies the attention mechanism to an RNN encoder-decoder.

[4] https://github.com/shawnwun/RNNLG.

- HLSTM proposed in [19] which uses a heuristic gate to ensure that all of the attribute-value information was accurately captured when generating.
- SCLSTM proposed in [22] which can learn the gating signal and language model jointly.

5 Results and Analysis

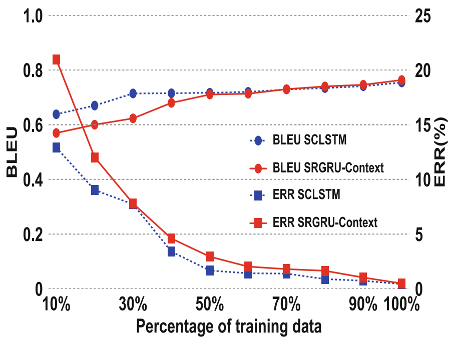
Overall, the proposed models SRGRUs consistently achieve better performance in term of the BLEU score in all domains. Especially, on the Hotel and TV datasets, the proposed models outperform the previous methods in both evaluation metrics. Moreover, our models also outperform the GRU basic model (GRU-Base) in all cases. However, the proposed models get worse on the Restaurant and Hotel datasets in terms of the error rate ERR score in comparison with SCLSTM. This indicates the advantage of the proposed refinement gate. A comparison of the two proposed generators is shown in Table 2: Without the backward RNN reranker, the generator tends to make semantic errors since it gains

Table 1. Comparison performance on four datasets in terms of the BLEU and the error rate ERR (%) scores; **bold** denotes the best and ***italic*** shows the second best model. The results were produced by training each network on 5 random initialization and selected model with the highest validation BLEU score.

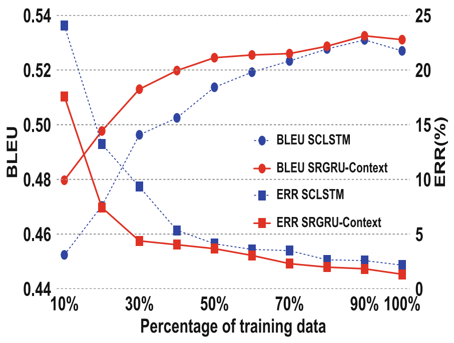
Model	Restaurant		Hotel		Laptop		TV	
	BLEU	ERR	BLEU	ERR	BLEU	ERR	BLEU	ERR
ENCDEC	0.7398	2.78%	0.8549	4.69%	0.5108	4.04%	0.5182	3.18%
HLSTM	0.7466	0.74%	0.8504	2.67%	0.5134	1.10%	0.5250	2.50%
SCLSTM	0.7525	**0.38%**	0.8482	3.07%	0.5116	**0.79%**	0.5265	2.31%
GRU-Base	0.7381	1.41%	0.8455	2.66%	0.5153	1.77%	0.5245	2.03%
SRGRU-Base	0.7549	0.56%	0.8640	***1.21***%	0.5190	1.56%	0.5305	1.62%
SRGRU-Context	***0.7634***	0.49%	**0.8776**	**0.98**%	***0.5191***	1.19%	***0.5311***	***1.33%***
TB-SRGRU	**0.7637**	***0.47***%	***0.8642***	1.56%	**0.5208**	***0.93***%	**0.5312**	**1.01%**

Table 2. Comparison performance on variety of SRGRU models on four datasets in terms of the BLEU and the error rate ERR(%) scores. The results were averaged over 5 randomly initialized networks for each proposed model. ^{a}reported in [22].

Model	Restaurant		Hotel		Laptop		TV	
	BLEU	ERR	BLEU	ERR	BLEU	ERR	BLEU	ERR
SCLSTM^a	0.7211	0.62%	0.8020	0.78%	-	-	-	-
+deep^a	0.7310	**0.46**%	0.8320	**0.41**%	-	-	-	-
GRU-Base	0.7208	1.55%	0.8426	1.97%	0.5158	1.94%	0.5244	2.11%
SRGRU-Base	0.7526	1.33%	0.8622	1.12%	0.5165	1.79%	0.5311	1.56%
SRGRU-Context	**0.7614**	0.99%	**0.8677**	1.75%	0.5182	1.41%	0.5312	1.37%
TB-SRGRU	0.7608	0.88%	0.8584	1.63%	**0.5188**	**1.35%**	**0.5316**	**1.27**%



(a) Curves on the Restaurant dataset

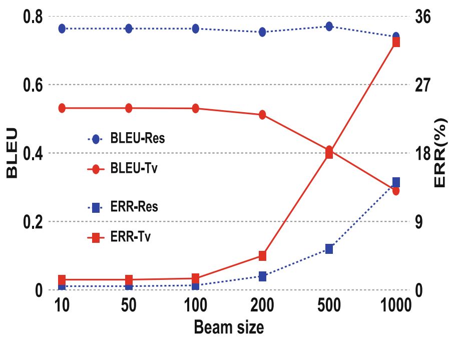


(b) Curves on the TV dataset

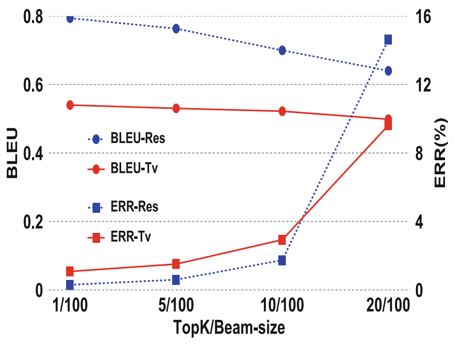
Fig. 2. Comparison of two generators SRGRU-Context and SCLSTM which are trained with different proportion of training data.

the higher slot error rate ERR. However, using the backward SRGRU reranker can improve the results in both evaluation metrics. This reranker provides benefit to the generator on producing higher-quality utterances.

Figure 2 compares two generators trained with different proportion of data evaluated on two metrics. As can be seen in Fig. 2a, the SCLSTM model achieves better results than SRGRU-Context model on both of BLEU and ERR scores since a small amount of training data was provided. However, the SRGRU-Context obtains the higher BLEU score and slightly higher ERR score as more training data was fed. On the other hand, in a more diverse dataset TV, the SRGRU-Context model consistently outperforms the SCLSTM on both evaluation metrics no matter how much training data is (Fig. 2b). This is mainly due to the ability of refinement gate which feeds to the GRU model a new input $\mathbf{x}_t$



(a) Curves on variation of Beam size



(b) Curves on variation of Top-k/Beam-size

Fig. 3. RGRU-Context generator was trained with different Beam size (a) and Top-k best results (b) and evaluated on Restaurant and TV datasets.

Table 3. Comparison of top responses generated for some input dialogue acts between different models. Errors are marked in color (missing, misplaced, repeated, grammar information). ^{a} and ^{b}denotes the baselines and the proposed models, respectively.

Model	Generated responses
Input DA	*inform_count(type = restaurant; count = 2; food = Basque; kidsallowed = no; pricerange = moderate)*
Reference	*There are 2 restaurants where no children are allowed in the moderate price range and serving Basque food*
ENCDEC^a	There are 2 restaurants that are moderately priced and do not allow kids. [Basque]
HLSTM^a	There are 2 Basque restaurants that are moderately priced and does not allow kids
SCLSTM^a	There are 2 Basque restaurants that are moderate and does not allow children
SRGRU-C^b	There are 2 moderate restaurants that serve Basque food and does not allow children
TB-SRGRU^b	There are 2 moderately priced Basque restaurants that do not allow kids
Input DA	*inform(name = the Carriage Inn; hasinternet = yes; dogsallowed = no)*
Reference	*The Carriage Inn is a nice place, it has internet and where no dogs are allowed*
ENCDEC^a	The Carriage Inn does not allow dogs. [has internet]
HLSTM^a	The Carriage Inn does not allow dogs, do not allow dogs [do not allow dogs]
SCLSTM^a	The Carriage Inn does not allow dogs and has internet
SRGRU-C^b	The Carriage Inn has internet and does not allow dogs
TB-SRGRU^b	The Carriage Inn does not allow dogs and it has internet
Input DA	*compare(name = Triton 52; ecorating = A+; family = L7; name = Hades 76; ecorating = C; family = L9)*
Reference	*Compared to Triton 52 which is in the A+ eco rating and is in the L7 product family, Hades 76 is in the C eco rating and is in the L9 product family. Which one do you prefer?*
ENCDEC^a	The Triton 52 has an A+ eco rating, the Hades 76 in the L7 product family and has an C eco rating. [L7, L9]
HLSTM^a	The Triton 52 is in the L7 product family with an A+ eco rating, while the Hades 76 has a C eco rating, which do you prefer? [L9]
SCLSTM^a	The Triton 52 has an A+ eco rating, the Hades 76 is in the L7 family and has a eco rating of C. [L7, L9]
SRGRU-C^b	The Triton 52 is in the L7 product family and an A+ eco rating, the Hades 76 is in the L9 family and has an C eco rating
TB-SRGRU^b	The Triton 52 has an A+ eco rating, in the L7 product family, the Hades 76 has a C eco rating and is in the L9 product family

conveying useful information filtered from the original input and the gating mechanism. Moreover, this gate also keeps the pattern of the generated utterance during generation. As a result, it can have a better realization of unseen slot-value pairs.

Figure 3a shows an effect of beam size on the SRGRU-Context model evaluated on Restaurant and TV datasets. As can be seen that, the model performs worse in terms of degrading the BLEU score and upgrading the slot error rate ERR when the beam size increases. The model seems to perform best with beam size less than 100. Figure 3b presents an effect of top-k best results in which we fixed the beam size at 100 and top-k best results varied as $k = 1$, 5, 10 and 20. In each case, the BLEU and the error rate ERR scores were computed on Restaurant and TV datasets. The results are consistent with Fig. 3a in which the BLEU and ERR scores get worse as more top-k best utterances were chosen.

Table 3 shows comparison of top responses generated by different models for given DAs. Firstly, both models SCLSTM and SRGRU-Context seem to produce the same kind of error, for example, *grammar* mistakes or *missing* information, partly because of using the same idea about gating mechanism. However, TB-SRGRU, with tying the refinement gate, has ability to fix this problem and produce the correct utterances (row 1 of Table 3). Secondly, as noted earlier, one problem of the previous methods is the ability to handle the *binary* slot and slots that take *don't_care* value. Both SCLSTM and the proposed models are able to handle this problem (row 2 of Table 3). Finally, the TV dataset is more diverse and much harder than the others because the order of slot-value pairs should be considered during generation. For example, to generate a comparison sentence of 2 items *Triton 52* and *Hades 76* for given dialogue act *compare(name = Triton 52; ecorating = A+; family = L7; name = Hades 76; ecorating = C; family = L9)*, the generator should consider that *A+* and *L7* values belong to the former item while *C* and *L9* values to the latter. The HLSTM tends to make the *repeated* information error while the ENCDEC and SCLSTM seem to *misplace* the slot value during generation. Take *L7* value, for instance, which should be generated follow the *Triton 52* instead of *Hades 76* as in row 3 of Table 3. We found that both proposed models SRGRU-Context and TB-SRGRU can deal with this problem to generate appropriate utterances.

6 Conclusion and Future Work

We propose a gating mechanism GRU-based generator, in which we introduced a refinement gate to semantically refine the original input tokens. The refined inputs conveying meaningful information are then fed into the GRU cell. The proposed models can learn from the unaligned data to produce natural language responses conditioned on the given DA. We extensively evaluated our model on four NLG datasets and compared against the previous generators. The results show that the proposed models obtain better performance than the existing generators on all of four NLG domains in terms of the BLEU and ERR metrics. In the future, we plan to further investigate the gating mechanism to multi-domain NLG since the refinement gate shows its ability to handle the unseen slot-value pairs.

Acknowledgment. This work was supported by the JSPS KAKENHI Grant number JP15K16048.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., et al.: Tensorflow: large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467 (2016)
2. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
3. Cheyer, A., Guzzoni, D.: Method and apparatus for building an intelligent automated assistant, US Patent 8,677,377, 18 March 2014
4. Dušek, O., Jurčíček, F.: Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. arXiv preprint arXiv:1606.05491 (2016)
5. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
6. Karpathy, A., Fei-Fei, L.: Deep visual-semantic alignments for generating image descriptions. In: Proceedings of CVPR, pp. 3128–3137 (2015)
7. Lowe, R., Pow, N., Serban, I., Charlin, L., Pineau, J.: Incorporating unstructured textual knowledge sources into neural dialogue systems. In: NIPS Workshop MLNLU (2015)
8. Mairesse, F., Young, S.: Stochastic language generation in dialogue using factored language models. Comput. Linguist. **40**(4), 763–799 (2014)
9. Mikolov, T.: Recurrent neural network based language model (2010)
10. Oh, A.H., Rudnicky, A.I.: Stochastic language generation for spoken dialogue systems. In: Proceedings of NAACL. ACL (2000)
11. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: BLEU: a method for automatic evaluation of machine translation. In: Proceedings of ACL, pp. 311–318. ACL (2002)
12. Pennington, J., Socher, R., Manning, C.D.: Glove: global vectors for word representation. In: EMNLP, vol. 14 (2014)
13. Ratnaparkhi, A.: Trainable methods for surface natural language generation. In: Proceedings of NAACL. ACL (2000)
14. Rieser, V., Lemon, O., Liu, X.: Optimising information presentation for spoken dialogue systems. In: Proceedings of ACL, pp. 1009-1018. ACL (2010)
15. Stent, A., Prasad, R., Walker, M.: Trainable sentence planning for complex information presentation in spoken dialog systems. In: Proceedings of ACL, p. 79. ACL (2004)
16. Tran, V.K., Nguyen, L.M.: Natural language generation for spoken dialogue system using RNN encoder-decoder networks. In: CoNLL 2017 (2017)
17. Vinyals, O., Toshev, A., Bengio, S., Erhan, D.: Show and tell: a neural image caption generator. In: CVPR (2015)
18. Wang, B., Liu, K., Zhao, J.: Inner attention based recurrent neural networks for answer selection (2016)
19. Wen, T.H., Gašić, M., Kim, D., Mrkšić, N., Su, P.H., Vandyke, D., Young, S.: Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. In: Proceedings of SIGDIAL. ACL (2015)
20. Wen, T.H., Gasic, M., Mrksic, N., Rojas-Barahona, L.M., Su, P.H., Vandyke, D., Young, S.: Multi-domain neural network language generation for spoken dialogue systems. arXiv preprint arXiv:1603.01232 (2016)

21. Wen, T.H., Gašic, M., Mrkšic, N., Rojas-Barahona, L.M., Su, P.H., Vandyke, D., Young, S.: Toward multi-domain language generation using recurrent neural networks (2016)
22. Wen, T.H., Gašić, M., Mrkšić, N., Su, P.H., Vandyke, D., Young, S.: Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In: Proceedings of EMNLP. ACL (2015)
23. Wen, T.H., Vandyke, D., Mrksic, N., Gasic, M., Rojas-Barahona, L.M., Su, P.H., Ultes, S., Young, S.: A network-based end-to-end trainable task-oriented dialogue system. arXiv preprint arXiv:1604.04562 (2016)
24. Zhang, X., Lapata, M.: Chinese poetry generation with recurrent neural networks. In: EMNLP, pp. 670–680 (2014)