Natural Language Generation for Spoken Dialogue System using RNN Encoder-Decoder Networks

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, Vietnam
tvkhanh@ictu.edu.vn

Abstract

Natural language generation (NLG) is a critical component in a spoken dialogue system. This paper presents a Recurrent Neural Network based Encoder-Decoder architecture, in which an LSTM-based decoder is introduced to select, aggregate semantic elements produced by an attention mechanism over the input elements, and to produce the required utterances. The proposed generator can be jointly trained both sentence planning and surface realization to produce natural language sentences. The proposed model was extensively evaluated on four different NLG datasets. The experimental results showed that the proposed generators not only consistently outperform the previous methods across all the NLG domains but also show an ability to generalize from a new, unseen domain and learn from multi-domain datasets.

1 Introduction

Natural Language Generation (NLG) plays a critical role in Spoken Dialogue Systems (SDS) with task is to convert a meaning representation produced by the Dialogue Manager into natural language utterances. Conventional approaches still rely on comprehensive hand-tuning templates and rules requiring expert knowledge of linguistic representation, including rule-based (Mirkovic et al., 2011), corpus-based n-gram models (Oh and Rudnicky, 2000), and a trainable generator (Stent et al., 2004).

Recently, Recurrent Neural Networks (RNNs) based approaches have shown promising performance in tackling the NLG problems. The RNN-based models have been applied for NLG as a joint training model (Wen et al., 2015a,b) and an end-to-end training model (Wen et al., 2016c). A recurring problem in such systems is requiring annotated datasets for particular dialogue acts[1] (DAs). To ensure that the generated utterance representing the intended meaning of the given DA, the previous RNN-based models were further conditioned on a 1-hot vector representation of the DA. Wen et al. (2015a) introduced a heuristic gate to ensure that all the slot-value pair was accurately captured during generation. Wen et al. (2015b) subsequently proposed a Semantically Conditioned Long Short-term Memory generator (SC-LSTM) which jointly learned the DA gating signal and language model.

More recently, Encoder-Decoder networks (**??**), especially the attentional based models (Wen et al., 2016b; Mei et al., 2015) have been explored to solve the NLG tasks. The Attentional RNN Encoder-Decoder (Bahdanau et al., 2014) (ARED) based approaches have also shown improved performance on a variety of tasks, e.g., image captioning (Xu et al., 2015; Yang et al., 2016), text summarization (Rush et al., 2015; Nallapati et al., 2016).

While the RNN-based generators with DA gating-vector can prevent the undesirable semantic repetitions, the ARED-based generators show signs of better adapting to a new domain. However, none of the models show significant advantage from out-of-domain data. To better analyze model generalization to an unseen, new domain as well as model leveraging the out-of-domain sources, we propose a new architecture which is an extension of the ARED model. In order to better select, aggregate and control the semantic information, a Refinement Adjustment LSTM-based component (*RALSTM*) is introduced

[1]A combination of an action type and a set of slot-value pairs. e.g. *inform(name='Bar crudo'; food='raw food')*

to the decoder side. The proposed model can learn from unaligned data by jointly training the sentence planning and surface realization to produce natural language sentences. We conducted experiments on four different NLG domains and found that the proposed methods significantly outperformed the state-of-the-art methods regarding BLEU (Papineni et al., 2002) and slot error rate ERR scores (Wen et al., 2015b). The results also showed that our generators could scale to new domains by leveraging the out-of-domain data. To sum up, we make three key contributions in this paper:

- We present an LSTM-based component called *RALSTM* cell applied on the decoder side of an ARED model, resulting in an end-to-end generator that empirically shows significant improved performances in comparison with the previous approaches.
- We extensively conduct the experiments to evaluate the models training from scratch on each in-domain dataset.
- We empirically assess the models' ability to: learn from multi-domain datasets by pooling all available training datasets; and adapt to a new, unseen domain by limited feeding amount of in-domain data.

We review related works in Section 2. Following a detail of proposed model in Section 3, Section 4 describes datasets, experimental setups, and evaluation metrics. The resulting analysis is presented in Section 5. We conclude with a brief summary and future work in Section 6.

2 Related Work

Recently, RNNs-based models have shown promising performance in tackling the NLG problems. Zhang and Lapata (2014) proposed a generator using RNNs to create Chinese poetry. Xu et al. (2015); Karpathy and Fei-Fei (2015); Vinyals et al. (2015) also used RNNs in a multi-modal setting to solve image captioning tasks. The RNN-based Sequence to Sequence models have applied to solve variety of tasks: conversational modeling (**???**), machine translation (**??**)

For task-oriented dialogue systems, Wen et al. (2015a) combined a forward RNN generator, a CNN reranker, and a backward RNN reranker to generate utterances. Wen et al. (2015b) proposed SC-LSTM generator which introduced a control sigmoid gate to the LSTM cell to jointly learn the gating mechanism and language model. A recurring problem in such systems is the lack of sufficient domain-specific annotated data. Wen et al. (2016a) proposed an out-of-domain model which was trained on counterfeited data by using semantically similar slots from the target domain instead of the slots belonging to the out-of-domain dataset. The results showed that the model can achieve a satisfactory performance with a small amount of in-domain data by fine tuning the target domain on the out-of-domain trained model.

More recently, RNN encoder-decoder based models with attention mechanism (Bahdanau et al., 2014) have shown improved performances in various tasks. Yang et al. (2016) proposed a review network to the image captioning, which reviews all the information encoded by the encoder and produces a compact thought vector. Mei et al. (2015) proposed RNN encoder-decoder-based model by using two attention layers to jointly train content selection and surface realization. More close to our work, Wen et al. (2016b) proposed an attentive encoder-decoder based generator which computed the attention mechanism over the slot-value pairs. The model showed a domain scalability when a very limited amount of data is available.

Moving from a limited domain dialogue system to an open domain dialogue system raises some issues. Therefore, it is important to build an open domain dialogue system that can make as much use of existing abilities of functioning from other domains. There have been several works to tackle this problem, such as (Mrkšić et al., 2015) using RNN-based networks for multi-domain dialogue state tracking, (Wen et al., 2016a) using a procedure to train multi-domain via multiple adaptation steps, or (Gašić et al., 2015; Williams, 2013) adapting of SDS components to new domains.

3 Recurrent Neural Language Generator

The recurrent language generator proposed in this paper is based on a neural language generator (Wen et al., 2016b), which consists of three main components: (i) an Encoder that incorporates the target meaning representation (MR) as the model inputs, (ii) an Aligner that aligns and controls the semantic elements, and (iii) an RNN Decoder that

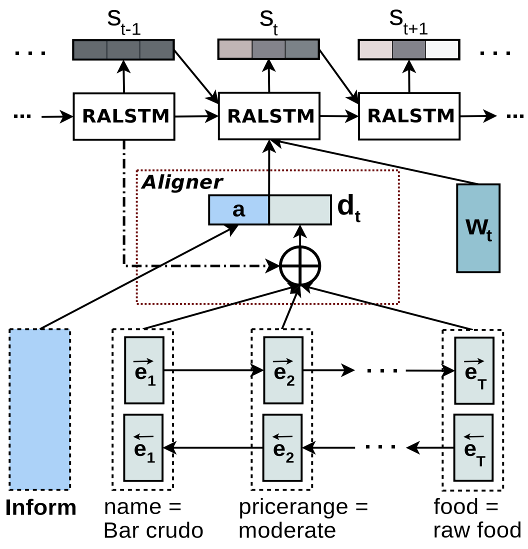


Figure 1: Unrolled presentation of the RNNs-based neural language generator. The Encoder part is a BiLSTM, the Aligner is an attention mechanism over the encoded inputs, and the Decoder is the proposed RALSTM model conditioned on a 1-hot representation vector **s**. The fading color of the vector **s** indicates retaining information for future computational time steps.

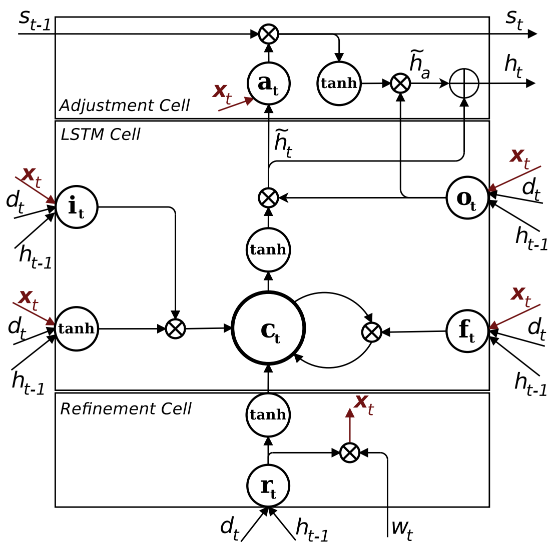


Figure 2: The RALSTM cell proposed in this paper, which consists of three components: an Refinement Cell, a traditional LSTM Cell, and an Adjustment Cell. At time step t, while the Refinement cell computes new input tokens $\mathbf{x}_t$ based on the original input tokens and the attentional DA representation $\mathbf{d}_t$, the Adjustment Cell calculates how much information of the slot-value pairs can be generated by the LSTM Cell.

generates output sentences. The generator architecture is shown in Figure 1. The Encoder first encodes the MR into input semantic elements which are then aggregated and selected by utilizing an attention-based mechanism by the Aligner. The input to the RNN Decoder at each time step is a 1-hot encoding of a token[2] $\mathbf{w}_t$ and an attentive DA representation $\mathbf{d}_t$. At each time step t, RNN Decoder also computes how much the feature value vector $\mathbf{s}_{t-1}$ retained for the next computational steps, and adds this information to the RNN output which represents the probability distribution of the next token $\mathbf{w}_{t+1}$. At generation time, we can sample from this conditional distribution to obtain the next token in a generated sentence, and feed it as the next input to the RNN Decoder. This process finishes when an end sign is generated (Karpathy and Fei-Fei, 2015), or some constraints are reached (Zhang and Lapata, 2014). The model can produce a sequence of tokens which can finally be lexicalized[3] to form the required utterance.

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

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

3.1 Encoder

The slots and values are separated parameters used in the encoder side. This embeds the source information into a vector representation $\mathbf{z}_i$ which is a concatenation of embedding vector representation of each slot-value pair, and is computed by:

$$\mathbf{z}_i = \mathbf{u}_i \oplus \mathbf{v}_i \qquad (1)$$

where $\mathbf{u}_i$, $\mathbf{v}_i$ are the i-th slot and value embedding vectors, respectively, and $\oplus$ is vector concatenation. The i index runs over the L given slot-value pairs. In this work, we use a 1-layer, Bidirectional LSTM (Bi-LSTM) to encode the sequence of slot-value pairs[4] embedding. The Bi-LSTM consists of forward and backward LSTMs which read the sequence of slot-value pairs from left-to-right and right-to-left to produce forward and backward sequence of hidden states $(\overrightarrow{\mathbf{e}_1}, .., \overrightarrow{\mathbf{e}_L})$, and $(\overleftarrow{\mathbf{e}_1}, .., \overleftarrow{\mathbf{e}_L})$, respectively. We then obtain the sequence of encoded hidden states $\mathbf{E} = (\mathbf{e}_1, \mathbf{e}_2, .., \mathbf{e}_L)$ where $\mathbf{e}_i$

[4]We treated the set of slot-value pairs as a sequence and use the order specified by slot's name (e.g., slot *address* comes first, *food* follows *address*). We have tried treating slot-value pairs as a set with natural order as in the given DAs. However, this yielded even worse results.

is a sum of the forward hidden state $\overrightarrow{\mathbf{e}_i}$ and the backward one $\overleftarrow{\mathbf{e}_i}$ as follows:

$$\mathbf{e}_i = \overrightarrow{\mathbf{e}_i} + \overleftarrow{\mathbf{e}_i} \tag{2}$$

3.2 Aligner

The Aligner utilizes attention mechanism to calculate the DA representation as follows:

$$\beta_{t,i} = \frac{\exp e_{t,i}}{\sum_j \exp e_{t,j}} \tag{3}$$

where

$$e_{t,i} = a(\mathbf{e}_i, \mathbf{h}_{t-1}) \tag{4}$$

and $\beta_{t,i}$ is the weight of i-th slot-value pair calculated by the attention mechanism. The alignment model a is computed by:

$$a(\mathbf{e}_i, \mathbf{h}_{t-1}) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a \mathbf{e}_i + \mathbf{U}_a \mathbf{h}_{t-1}) \tag{5}$$

where $\mathbf{v}_a, \mathbf{W}_a, \mathbf{U}_a$ are the weight matrices to learn. Finally, the Aligner calculates dialogue act embedding $\mathbf{d}_t$ as follows:

$$\mathbf{d}_t = \mathbf{a} \oplus \sum_i \beta_{t,i} \mathbf{e}_i \tag{6}$$

where $\mathbf{a}$ is vector embedding of the action type.

3.3 RALSTM Decoder

The proposed semantic RALSTM cell applied for Decoder side consists of three components: a Refinement cell, a traditional LSTM cell, and an Adjustment cell:

Firstly, instead of feeding the original input token $\mathbf{w}_t$ into the RNN cell, the input is recomputed by using a semantic gate as follows:

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

where $\mathbf{W}_{rd}$ and $\mathbf{W}_{rh}$ are weight matrices. Element-wise multiplication $\odot$ plays a part in word-level matching which not only learns the vector similarity, but also preserves information about the two vectors. $\mathbf{W}_{rh}$ 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 DA representation $\mathbf{d}_t$, and the hidden context $\mathbf{h}_{t-1}$. $\mathbf{r}_t$ is called a *Refinement* gate because the input tokens are refined by a combination gating information of the attentive DA representation $\mathbf{d}_t$ and the previous hidden state $\mathbf{h}_{t-1}$. By this way, we can represent the whole sentence based on the refined inputs.

Secondly, the traditional LSTM network proposed by Hochreiter and Schmidhuber (2014) in which the input gate $\mathbf{i}_i$, forget gate $\mathbf{f}_t$ and output gates $\mathbf{o}_t$ are introduced to control information flow and computed as follows:

$$\begin{pmatrix}\mathbf{i}_t\\ \mathbf{f}_t\\ \mathbf{o}_t\\ \hat{\mathbf{c}}_t\end{pmatrix} = \begin{pmatrix}\sigma\\ \sigma\\ \sigma\\ \tanh\end{pmatrix} \mathbf{W}_{4n,4n} \begin{pmatrix}\mathbf{x}_t\\ \mathbf{d}_t\\ \mathbf{h}_{t-1}\end{pmatrix} \tag{8}$$

where n is hidden layer size, $\mathbf{W}_{4n,4n}$ is model parameters. The cell memory value $\mathbf{c}_t$ is modified to depend on the DA representation as:

$$\begin{aligned}\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \hat{\mathbf{c}}_t + \tanh(\mathbf{W}_{cr}\mathbf{r}_t)\\ \tilde{\mathbf{h}}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)\end{aligned} \tag{9}$$

where $\tilde{\mathbf{h}}_t$ is the output.

Thirdly, inspired by work of Wen et al. (2015b) in which the generator was further conditioned on a 1-hot representation vector $\mathbf{s}$ of given dialogue act, and work of Lu et al. (2016) that proposed a visual sentinel gate to make a decision on whether the model should attend to the image or to the sentinel gate, an additional gating cell is introduced on top of the traditional LSTM to gate another controlling vector $\mathbf{s}$. Figure 6 shows how RALSTM controls the DA vector $\mathbf{s}$. First, starting from the 1-hot vector of the DA $\mathbf{s}_0$, at each time step t the proposed cell computes how much the LSTM output $\tilde{\mathbf{h}}_t$ affects the DA vector, which is computed as follows:

$$\begin{aligned}\mathbf{a}_t &= \sigma(\mathbf{W}_{ax}\mathbf{x}_t + \mathbf{W}_{ah}\tilde{\mathbf{h}}_t)\\ \mathbf{s}_t &= \mathbf{s}_{t-1} \odot \mathbf{a}_t\end{aligned} \tag{10}$$

where $\mathbf{W}_{ax}$, $\mathbf{W}_{ah}$ are weight matrices to be learned. $\mathbf{a}_t$ is called an *Adjustment* gate since its task is to control what information of the given DA have been generated and what information should be retained for future time steps. Second, we consider how much the information preserved in the DA $\mathbf{s}_t$ can be contributed to the output, in which an additional output is computed by applying the output gate $\mathbf{o}_t$ on the remaining information in $\mathbf{s}_t$ as follows:

$$\begin{aligned}\mathbf{c}_a &= \sigma(\mathbf{W}_{os}\mathbf{s}_t)\\ \tilde{\mathbf{h}}_a &= \mathbf{o}_t \odot \tanh(\mathbf{c}_a)\end{aligned} \tag{11}$$

where $\mathbf{W}_{os}$ is a weight matrix to project the DA presentation into the output space, $\tilde{\mathbf{h}}_a$ is the Adjustment cell output. Final RALSTM output is a combination of both outputs of the traditional LSTM cell and the Adjustment cell, and computed as follows:

$$\mathbf{h}_t = \tilde{\mathbf{h}}_t + \tilde{\mathbf{h}}_a \tag{12}$$

Finally, the output distribution is computed by applying a softmax function g, and the distribution can be sampled to obtain the next token,

$$\begin{aligned} P(w_{t+1} \mid w_t, ...w_0, \mathbf{DA}) &= g(\mathbf{W}_{ho}\mathbf{h}_t) \\ w_{t+1} &\sim P(w_{t+1} \mid w_t, w_{t-1}, ...w_0, \mathbf{DA}) \end{aligned} \tag{13}$$

where $\mathbf{DA} = (\mathbf{s}, \mathbf{z})$.

3.4 Training

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

$$\mathbf{F}(\theta) = -\sum_{t=1}^{T} \mathbf{y}_t^\top \log \mathbf{p}_t \tag{14}$$

where: $\mathbf{y}_t$ is the ground truth token distribution, $\mathbf{p}_t$ is the predicted token distribution, T is length of the input sentence. The proposed generators were trained by treating each sentence as a mini-batch with l_2 regularization added to the objective function for every 5 training examples. The models were initialized with a pretrained Glove word embedding vectors (Pennington et al., 2014) and optimized by using stochastic gradient descent and back propagation through time (Werbos, 1990). Early stopping mechanism was implemented to prevent over-fitting by using a validation set as suggested in (Mikolov, 2010).

3.5 Decoding

The decoding consists of two phases: (i) over-generation, and (ii) reranking. In the over-generation, the generator conditioned on both representations of the given DA use a beam search to generate a set of candidate responses. In the reranking phase, cost of the generator is computed to form the reranking score $\mathbf{R}$ as follows:

$$\mathbf{R} = \mathbf{F}(\theta) + \lambda \mathbf{ERR} \tag{15}$$

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

$$\mathbf{ERR} = \frac{\mathbf{p} + \mathbf{q}}{\mathbf{N}} \tag{16}$$

where $\mathbf{N}$ is the total number of slots in DA, and $\mathbf{p}$, $\mathbf{q}$ is the number of missing and redundant slots, respectively.

4 Experiments

We extensively conducted a set of experiments to assess the effectiveness of the proposed models by using several metrics, datasets, and model architectures, in order to compare to prior methods.

4.1 Datasets

We assessed the proposed models on four different NLG domains: finding a restaurant, finding a hotel, buying a laptop, and buying a television. The Restaurant and Hotel were collected in (Wen et al., 2015b), while the Laptop and TV datasets have been released by (Wen et al., 2016a) with 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. This makes the NLG tasks for the Laptop and TV domains become much harder. The dataset statistics are shown in Table 1.

Table 1: Dataset statistics.

	Restaurant	**Hotel**	**Laptop**	**TV**
# train	3,114	3,223	7,944	4,221
# validation	1,039	1,075	2,649	1,407
# test	1,039	1,075	2,649	1,407
# distinct DAs	248	164	13,242	7,035
# DA types	8	8	14	14
# slots	12	12	19	15

4.2 Experimental Setups

The generators were implemented using the TensorFlow library (Abadi et al., 2016) and trained with training, validation and testing ratio as 3:1:1. The hidden layer size, beam size were set to be 80 and 10, respectively, and the generators were trained with a 70% of dropout rate. We performed 5 runs with different random initialization of the network and the training is terminated by using early stopping. We then chose a model that yields the highest BLEU score on the validation set as shown in Table 2. Since the trained models can

Table 2: Performance comparison on four datasets in terms of the BLEU and the error rate ERR(%) scores. The results were produced by training each network on 5 random initialization and selected model with the highest validation BLEU score. $\sharp$ denotes the Attention-based Encoder-Decoder model. The best and second best models highlighted in **bold** and ***italic*** face, respectively.

Model	Restaurant		Hotel		Laptop		TV	
	BLEU	ERR	BLEU	ERR	BLEU	ERR	BLEU	ERR
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%
Enc-Dec$^\sharp$	0.7398	2.78%	0.8549	4.69%	0.5108	4.04%	0.5182	3.18%
w/o A$^\sharp$	0.7651	0.99%	0.8940	1.82%	0.5219	1.64%	0.5296	2.40%
w/o R$^\sharp$	***0.7748***	***0.22%***	***0.8944***	***0.48%***	***0.5235***	***0.57%***	***0.5350***	***0.72%***
RALSTM$^\sharp$	**0.7789**	**0.16%**	**0.8981**	**0.43%**	**0.5252**	**0.42%**	**0.5406**	**0.63%**

Table 3: Performance comparison of the proposed models on four datasets in terms of the BLEU and the error rate ERR(%) scores. The results were averaged over 5 randomly initialized networks. **bold** denotes the best model.

Model	Restaurant		Hotel		Laptop		TV	
	BLEU	ERR	BLEU	ERR	BLEU	ERR	BLEU	ERR
w/o A	0.7619	2.26%	0.8913	1.85%	0.5180	1.81%	0.5270	2.10%
w/o R	0.7733	0.23%	0.8901	0.59%	0.5208	0.60%	0.5321	0.50%
RALSTM	**0.7779**	**0.20%**	**0.8965**	**0.58%**	**0.5231**	**0.50%**	**0.5373**	**0.49%**

differ depending on the initialization, we also report the results which were averaged over 5 randomly initialized networks. Note that, except the results reported in Table 2, all the results shown were averaged over 5 randomly initialized networks. 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 sentences and selected the top 5 realizations after reranking. Moreover, in order to better understand the effectiveness of our proposed methods, we: (i) performed an ablation experiments to demonstrate the contribution of each proposed cells (Tables 2, 3), (ii) trained the models on the Laptop domain with varied proportion of training data, starting from 10% to 100% (Figure 3), (iii) trained general models by merging all the data from four domains together and tested them in each individual domain (Figure 4), and (iv) trained adaptation models on merging data from restaurant and hotel domains, then fine tuned the model on laptop domain with varied amount of adaptation data (Figure 5).

4.3 Evaluation Metrics and Baselines

The generator performance was assessed on the two evaluation metrics: the BLEU and the slot error rate ERR by adopting code from an open source benchmark toolkit for Natural Language Generation[5]. We compared the proposed models against three strong baselines which have been recently published as state-of-the-art NLG benchmarks[5].

- HLSTM proposed by Wen et al. (2015a) which used a heuristic gate to ensure that all of the slot-value information was accurately captured when generating.
- SCLSTM proposed by Wen et al. (2015b) which can jointly learn the gating signal and language model.
- Enc-Dec proposed by Wen et al. (2016b) which applied the attention-based encoder-decoder architecture.

5 Results and Analysis

5.1 Results

We conducted extensive experiments on our models and compared against the previous methods. Overall, the proposed models consistently achieve the better performance regarding both evaluation metrics across all domains in all test cases.

Model Comparison in an Unseen Domain

The ablation studies (Tables 2, 3) demonstrate the contribution of different model components

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

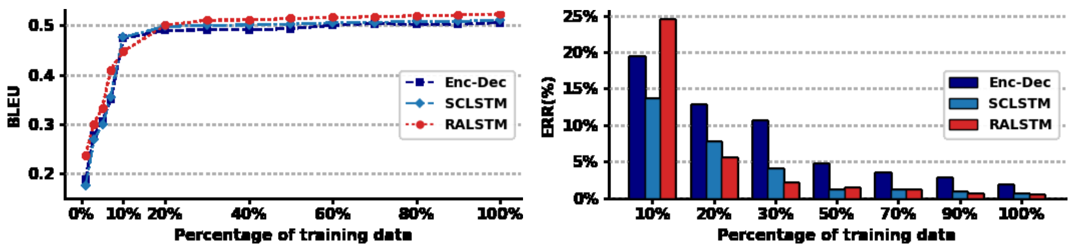


Figure 3: Performance comparison of the models trained on Laptop domain.

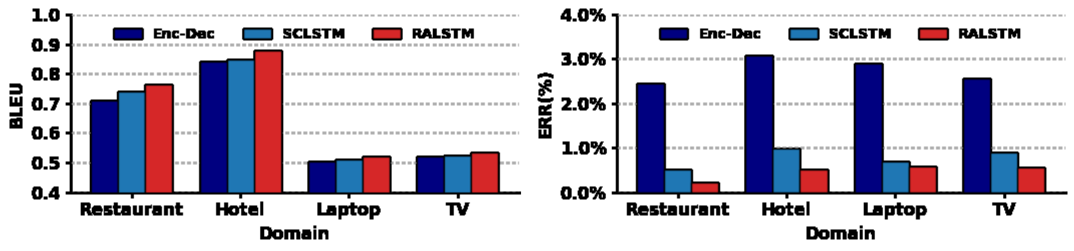


Figure 4: Performance comparison of the general models on four different domains.

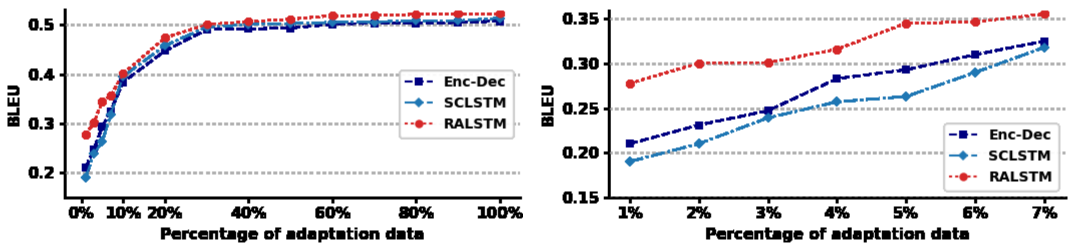


Figure 5: Performance on Laptop domain with varied amount of the adaptation training data when adapting models trained on Restaurant+Hotel dataset.

in which the models were assessed without Adjustment cell (*w/o A*), or without Refinement cell (*w/o R*). It clearly sees that the Adjustment cell contributes to reducing the slot error rate ERR score since it can effectively prevent the undesirable slot-value pair repetitions by gating the DA vector **s**. A comparison between the ARED-based models (denoted by $^\sharp$ in Table 2) shows that the proposed models not only have better performance with higher the BLEU score but also significantly reduce the slot error rate ERR score by a large margin about 2% to 4% in every datasets. Moreover, a comparison between the models with gating the DA vector also indicates that the proposed models (*w/o R*, *RALSTM*) have significant improved performance on both the evaluation metrics across the four domains compared to the *SCLSTM* model. The *RALSTM* cell without the Refinement cell is similar as the *SCLSTM* cell. However, it obtained the results much better than the SCLSTM baselines. This stipulates the necessary of the LSTM encoder and the Aligner in effectively partial learning the correlated order between slot-value representation in the DAs, especially for the unseen domain where there is only one training example for each DA. Table 3 further demonstrates the stable strength of our models since the results' pattern stays unchanged compared to those in Table 2.

Figure 3 shows a comparison of three models (*Enc-Dec*, *SCLSTM*, and *RALSTM*) which were trained from scratch on the unseen laptop domain in varied proportion of training data, from 1% to 100%. It clearly shows that the *RALSTM* outperforms the previous models in all cases, while the *Enc-Dec* has a much greater ERR score comparing to the two models.

A comparison of top responses generated for some input DAs between different models are shown in Table 4. While the previous models still

Table 4: Comparison of top responses generated for some input dialogue acts between different models. Errors are marked in color (missing, misplaced information). *All2** are general models.

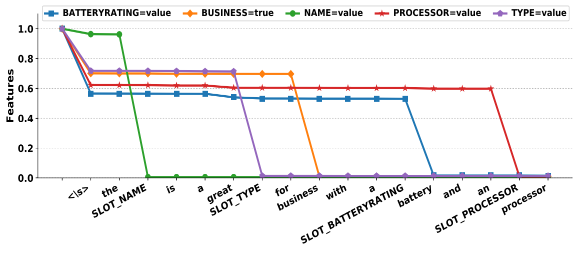
#	Model	Generated Responses from Laptop Domain
1	***Input DA***	?compare(name=”satellite notus 19”, pricerange=”budget”, drive=”500 gb”, name=”portege thanatos 98”, pricerange=”expensive”, drive=”128 gb”)
	Reference	the satellite notus 19 has a 500 gb drive and is in the budget price range . on the other hand the portege thanatos 98 has a 128 gb drive and is in the expensive price range . which would you prefer
	HLSTM	the satellite notus 19 is a budget priced laptop with a 500 gb drive and a 128 gb drive . which one do you prefer [portege thanatos 98, expensive]
	Enc-Dec	the satellite notus 19 is in the budget price range . the portege thanatos 98 has a 500 gb drive and is in the expensive price range . which one do you prefer [128 gb]
	SCLSTM	the satellite notus 19 is in the budget price range and has a 500 gb drive and the portege thanatos 98 has a 128 gb drive . which one do you prefer [expensive]
	RALSTM	the satellite notus 19 is in the budget price range with a 500 gb drive , while the portege thanatos 98 is in the expensive price range and has a 128 gb drive . which one do you prefer
	All2Laptop	the satellite notus 19 is a budget price range with a 500 gb drive and the portege thanatos 98 has a 128 gb drive and is in the expensive price range . which one do you prefer
#	**Model**	**Generated Responses from TV Domain**
2	***Input DA***	inform_count(count=”73”, type=”television”, hasusbport=”dontcare”, hdmiport=”2”, screensizerange=”dontcare”)
	Reference	there are 73 televisions if you do not care about whether it has usb port or not, with 2 hdmi ports and if you do not care about the screen size
	HLSTM	there are 73 televisions with 2 hdmi ports and if you do not care about the number of usb ports [screensizerange]
	Enc-Dec	there are 73 televisions with 2 hdmi ports if you do not care about usb ports [screensizerange]
	SCLSTM	there are 73 televisions with 2 hdmi ports if you do not care about the screen size and if you do not care about the number of usb ports
	RALSTM	there are 73 televisions with 2 hdmi ports if you do not care about screen size or usb ports
	All2Tv	there are 73 televisions with 2 hdmi ports if you do not care about screen size or usb ports

produce some errors (missing and misplaced information), the proposed models (RALSTM and the models *All2** trained by pooling all datasets together) can generate appropriate sentences. We also found that the proposed models tend to generate more complete and concise sentences than the other models.

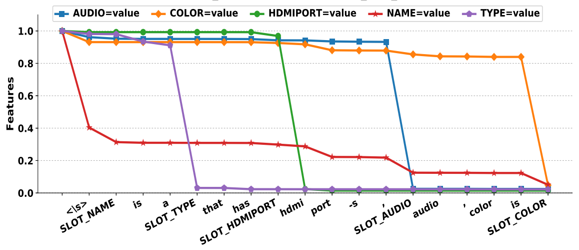
All these prove the importance of the proposed components: the Refinement cell in aggregating and selecting the attentive information, and the Adjustment cell in controlling the feature vector (see Examples in Figure 6).

General Models

Figure 4 shows a comparison performance of general models as described in Section 4.2. The results are consistent with the Figure 3, in which the *RALSTM* has better performance than the *Enc-Dec* and *SCLSTM* on all domains in terms of the BLEU and the ERR scores, while the *Enc-Dec* has difficulties in reducing the ERR score. This indicates the relevant contribution of the proposed component Refinement and Adjustment cells to the original ARED architecture, in which the Refinement with attentional gating can effectively select and aggregate the information before putting them into the traditional LSTM cell, while the Adjustment with gating DA vector can effectively control the information flow during generation.



(a) An example from the Laptop domain.



(b) An example from the TV domain.

Figure 6: Example showing how RALSTM drives down the DA feature value vector **s** step-by-step, in which the model generally shows its ability to detect words and phases describing a corresponding slot-value pair.

Adaptation Models

Figure 5 shows domain scalability of the three models in which the models were first trained on the merging out-of-domain Restaurant and Hotel datasets, then fine tuned the parameters with varied amount of in-domain training data (laptop domain). The *RALSTM* model outperforms the previous model in both cases where the sufficient in-domain data is used (as in Figure 5-*left*) and the limited in-domain data is used (Figure 5-*right*). The Figure 5-*right* also indicates that the *RALSTM* model can adapt to a new, unseen domain faster than the previous models.

6 Conclusion and Future Work

We present an extension of ARED model, in which an RALSTM component is introduced to select and aggregate semantic elements produced by the Encoder, and to generate the required sentence. We assessed the proposed models on four NLG domains and compared to the state-of-the-art generators. The proposed models empirically show consistent improvement over the previous methods in both the BLEU and ERR evaluation metrics. The proposed models also show an ability to extend to a new, unseen domain no matter how much the in-domain training data was fed. In the future, it would be interesting to apply the proposed model to other tasks that can be modeled based on the encoder-decoder architecture, i.e., image captioning, reading comprehension, and machine translation.

References

Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Milica Gašić, Dongho Kim, Pirros Tsiakoulis, and Steve Young. 2015. Distributed dialogue policies for multi-domain statistical dialogue management. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, pages 5371–5375.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* .

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference CVPR*. pages 3128–3137.

Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. 2016. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. *arXiv preprint arXiv:1612.01887* .

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2015. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. *arXiv preprint arXiv:1509.00838* .

Tomas Mikolov. 2010. Recurrent neural network based language model.

Danilo Mirkovic, Lawrence Cavedon, Matthew Purver, Florin Ratiu, Tobias Scheideck, Fuliang Weng, Qi Zhang, and Kui Xu. 2011. Dialogue management using scripts and combined confidence scores. US Patent 7,904,297.

Nikola Mrkšić, Diarmuid O Séaghdha, Blaise Thomson, Milica Gašić, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2015. Multi-domain dialog state tracking using recurrent neural networks. *arXiv preprint arXiv:1506.07190* .

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023* .

Alice H Oh and Alexander I Rudnicky. 2000. Stochastic language generation for spoken dialogue systems. In *Proceedings of the 2000 ANLP/NAACL Workshop on Conversational systems-Volume 3*. Association for Computational Linguistics, pages 27–32.

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 ACL*. Association for Computational Linguistics, pages 311–318.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. volume 14, pages 1532–43.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685* .

Amanda Stent, Rashmi Prasad, and Marilyn Walker. 2004. Trainable sentence planning for complex information presentation in spoken dialog systems. In *Proceedings of the 42nd ACL*. Association for Computational Linguistics, page 79.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 3156–3164.

Tsung-Hsien Wen, Milica Gašić, Dongho Kim, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015a. Stochastic Language Generation in Dialogue using Recurrent Neural Networks with Convolutional Sentence Reranking. In *Proceedings SIGDIAL*. Association for Computational Linguistics.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina M Rojas-Barahona, Pei-Hao Su, David Vandyke, and Steve Young. 2016a. Multi-domain neural network language generation for spoken dialogue systems. *arXiv preprint arXiv:1603.01232* .

Tsung-Hsien Wen, Milica Gašic, Nikola Mrkšic, Lina M Rojas-Barahona, Pei-Hao Su, David Vandyke, and Steve Young. 2016b. Toward multi-domain language generation using recurrent neural networks .

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015b. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of EMNLP*. Association for Computational Linguistics.

Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2016c. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562* .

Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE* 78(10):1550–1560.

Jason Williams. 2013. Multi-domain learning and generalization in dialog state tracking. In *Proceedings of SIGDIAL*. Citeseer, volume 62.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*. volume 14, pages 77–81.

Zhilin Yang, Ye Yuan, Yuexin Wu, William W Cohen, and Ruslan R Salakhutdinov. 2016. Review networks for caption generation. In *Advances in Neural Information Processing Systems*. pages 2361–2369.

Xingxing Zhang and Mirella Lapata. 2014. Chinese poetry generation with recurrent neural networks. In *EMNLP*. pages 670–680.