Self-Learning Architecture for Natural Language Generation

Hyungtak Choi, Siddarth K. M., Haehun Yang, Heesik Jeon, Inchul Hwang, Jihie Kim
Samsung Research, Samsung Electronics Co. Ltd., Seoul, Korea
{ht777.choi, siddarth.km, haehun.yang, heesik.jeon,
inc.hwang, jihie.kim}@samsung.com

Abstract

In this paper, we propose a self-learning architecture for generating natural language templates for conversational assistants. Generating templates to cover all the combinations of slots in an intent is time consuming and labor-intensive. We examine three different models based on our proposed architecture - Rule-based model, Sequence-to-Sequence (Seq2Seq) model and Semantically Conditioned LSTM (SC-LSTM) model for the IoT domain - to reduce the human labor required for template generation. We demonstrate the feasibility of template generation for the IoT domain using our self-learning architecture. In both automatic and human evaluation, the self-learning architecture outperforms previous works trained with a fully human-labeled dataset. This is promising for commercial conversational assistant solutions.

1 Introduction

Intelligent Conversational Assistants are prevalent now. They have been integrated into a wide range of IoT devices. Various recent studies on stochastic language generation have been conducted in NLG. Although these stochastic approaches outperform traditional LSTM language models, they have drawbacks including the requirement of large training datasets, low accuracy of trained models and lack of naturalness of generated sentences. Therefore, Most of commercial conversational assistant services adopt template-based approach (Cheyer and Guzzoni, 2014; Mirkovic and Cavedon, 2011) to implement natural language generation. This approach is robust and feasible for commercialization, but requires creating a large number of templates. In this approach, one needs to manually generate NLG templates that cover all the possible combinations of intents and slots.[1] There are statistical approaches for generating templates with 2, 3 and 4 slots (Narayan et al., 2011), but they suffer from exponential complexity as the number of slots increases. In addition, substitution-based implementation with SimpleNLG (Gatt and Reiter, 2009) can handle some of the cases, but this is not versatile enough. The number of templates required to cover all possible combinations of slots is:

$$\text{No. Templates}_n = 2^n - 1 \quad (1)$$

where n is the total number of possible slots. The value is exponential as shown in equation 1. Manually generating an exponential number of templates is undesirable, especially when the intents get more complex.

We propose a self-learning architecture for NLG which solves the problem of generating an exponential number of templates. In order to generate informative and natural sentences, arguably, it is more important to generate consistent system responses with limited syntactic information than to generate error prone system responses with more variation in their grammatical form. In our proposed solution, we start with an initial training set containing less than or equal to 2 slots per intent, and iteratively build our model to increase its ability to cover more complex inputs. Thus, the re-

[1] Throughout this paper, **intent** denotes the intention of user utterance and **slot** denotes the variable part (slot value) and its name if any (slot name). Slot can be replaced by another phrase in user utterances or responses. NLG will be generated from **dialog act** and it will be written as $Intent(SlotName1 = SlotValue1; SlotName2 = SlotValue2; \cdots)$. For example, from dialog act up(functionname=temperature;devicename='airconditioner';location='bedroom'), response "I turned up the temperature of the airconditioner in the bedroom" is generated.

Proceedings of The 11th International Natural Language Generation Conference, pages 165–170,
Tilburg, The Netherlands, November 5-8, 2018.

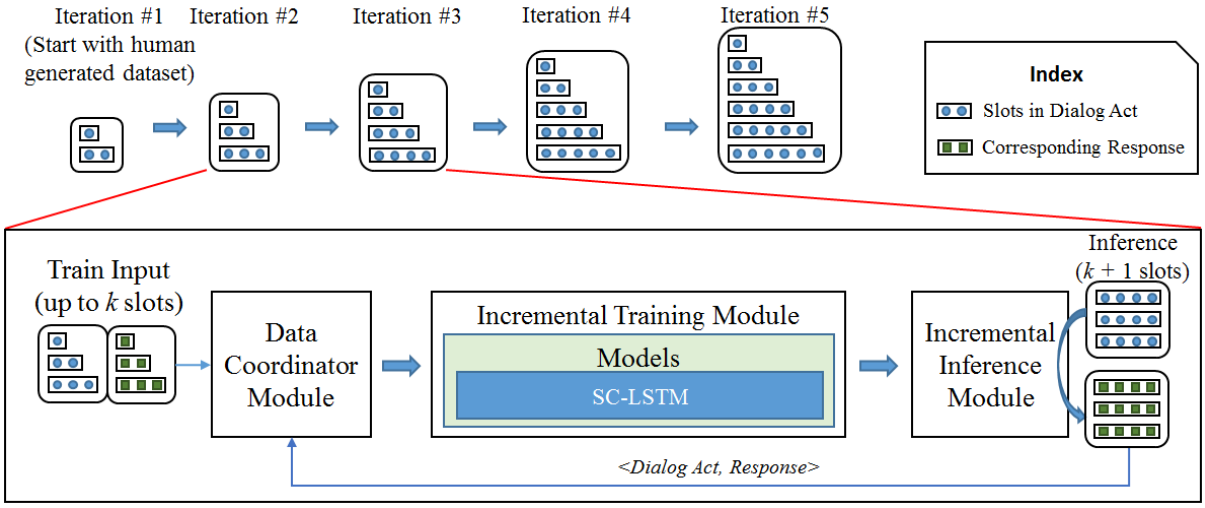

Figure 1: Proposed Self-Learning Architecture for NLG.

quired number of human generated templates decreases from $2^n - 1$ to $\binom{n}{1} + \binom{n}{2} = O(n^2)$.

2 Related Work

Several approaches have been studied for NLG systems. The template-based approach (McRoy et al., 2003; Channarukul et al., 2002) is most commonly used, because it guarantees the advantage of completely controlling the output quality. However, that approach has two main disadvantages. First, it is labor-intensive to generate and maintain templates (Galley et al., 2001). Second, it does not scale well when domains are changed or expanded (Channarukul et al., 2002; Reiter, 1996). Recently, various stochastic NLG methods such as Sequence to Sequence generation with attention (Seq2Seq w/ attn) (Dušek and Jurčíček, 2016) and Semantically Conditioned LSTM (SC-LSTM) (Wen et al., 2015) have been studied to overcome the disadvantages of template-based approaches. They also aim to remove the need for manual alignment of training dataset. They are directly trained on a data corpus and reduce the human effort required for producing templates. Other End-to-End deep learning approaches have been studied (Gehrmann et al., 2018), along with domain adaptation (Dethlefs, 2017), to solve the problem of domain specificity. Though they produce state-of-the-art results among stochastic approaches, the generated sentences are not natural enough for real-world conversational assistant services, where even the slightest mistakes can be detrimental. Another hurdle while training these models is the issue of finding the right dataset for training (Gatt and Krahmer, 2018). We show that our proposed self-learning architecture can outperform existing neural models and greatly reduce human effort compared to template-based approaches without compromising much on output quality.

3 Self-Learning Architecture for NLG

In this section, we explain the proposed architecture to resolve the problem discussed above. We also briefly explain the models that we use for response generation, namely, Rule-based, Seq2Seq and SC-LSTM. Initially, all dialog acts in the training dataset contain all the combinations of slots with less than k^2 slots per training instance.

As a preparation step, we manually generate the initial training dataset as $\langle DialogAct, Response \rangle$ pairs for all the combinations of slots with up to two slots per instance. With the initial value of k as 2, see Algorithm 1.

Thus, we have successfully increased the size of our training dataset. Sample data for successive training steps is shown in Table 1. We experimented with three different models for response generation task using Dialog Acts: Rule-based model, Seq2Seq model and SC-LSTM model.

Figure 1 depicts the flow for the self-learning architecture.

3.1 Rule-Based Approach

This method was inspired by (Filippova, 2010), where shortest path finding algorithms are used

[2]All references to k in this paper denote the number of slots present in a training instance

Algorithm 1 Self Learning Architecture

while $k <$ max_slots **do**
    Train a new language generation model that generates responses from dialog acts containing up to k slots
    Generate responses containing $k+1$ slots using Dialog Acts containing $k+1$ slots and the model trained above
    Augment the training dataset with the inferred sentences containing $k+1$ slots from the previous step
    $k = k + 1$
end while

Algorithm 2 Rule-Based Generation

procedure PROTO_RULE_BASED(A, i, j)
    $A1 =$ GET_UTTERANCE($A \setminus A[i]$)
    $A2 =$ GET_UTTERANCE($A \setminus A[j]$)
    return $SCS(A1, A2)$
end procedure
procedure RULE_BASED(A)
    for $(0 \leq i < j < |A|)$ **do**
        $Y_{i,j} =$ PROTO_RULE_BASED(A, i, j)
    end for
    $\widehat{Y} = \arg\min_{Y_{i,j}}\{\text{NUM\_WORDS}(Y_{i,j})\}$
    return $\widehat{Y}$
end procedure

to summarize multiple sentences. Although it is straightforward to understand and gives good scores on metrics such as longest common subsequence (LCS) metric used by (Zhao et al., 2002), the approach is different from our objective. We need to conserve all information from the source sentence. Consequently, we use the shortest common supersequence (SCS) for our task. Finding SCS of more than two sentences is an NP-complete problem (Räihä and Ukkonen, 1981), therefore we simplify it by considering SCS of two sentences.

Assumption 1: If sentence X is a good response for slot combination A and sentence Y is one for combination B, then their SCS Z will be a good response for slot combination $A \cup B$.
An example of A, B, X and Y is
A : up(functionname=temperature;devicename='airconditioner'),
B : up(functionname=temperature;location='bedroom'),
X : I will turn up the temperature for the airconditioner,
Y : I will turn up the temperature in the bedroom.
The SCS of X and Y is "I will turn up the temperature for the airconditioner in the bedroom", which is plausible response sentence for $A \cup B$.

Assumption 2: SCS Z will be an even better response if $|A \setminus B|$ and $|B \setminus A|$ are small.
Based on Assumptions 1 and 2 above, it is natural to think of a simple algorithm (see Algorithm 2) which generates natural response given slot combination A. There can be numerous candidates, so we opt for the shortest one.

3.2 Sequence to Sequence with Attention

Seq2Seq model (Sutskever et al., 2014) has been widely used in many machine learning tasks. For our task, we use an LSTM-based encoder and decoder model with Attention mechanism (Dušek and Jurčíček, 2016). The LSTM encoder takes a sequence of Dialog Acts as input, where each slot is a symbol in the vocabulary. While decoding, they use an LSTM-based re-ranker at the end to calculate slot errors and penalize responses that have wrong slots as compared to the input Dialog Act.

3.3 Semantically Conditioned LSTM

The SC-LSTM Model (Wen et al., 2015) deals with the issue of repetitive word generation in LSTM-based NLG. It receives the Dialog Act in the form of a bit vector, where each bit in the vector denotes whether a particular slot-value pair exists in the Dialog Act. This model tries to mitigate the issue by using additional control cell along with the standard LSTM cell. The control cell produces a surface realization which accurately encodes the input information and helps in cutting off repetitive words.

4 Experiments and Results

4.1 Dataset for Initial Training

We use a small domain-specific dataset for our experiment. Our dataset is derived from the one used by (Georgila et al., 2018), focused on the domain of IoT Home Appliances. We extract a subset of the system responses, and extend them for increased coverage, and generate $\langle DialogAct, Response\rangle$ pairs. The generated $\langle DialogAct, Response\rangle$ pairs are given to our

Phase	Data	# of slots	Sample training dataset
Phase 1	Training data	1	Dialog Act: up(functionname=temperature;devicename='airconditioner') Response: I turned up the temperature of the airconditioner
		2	Dialog Act: up(functionname=temperature;devicename='airconditioner';location='bedroom') Response: I turned up the temperature of the airconditioner in the bedroom
	Test data	3	Dialog Act: up(functionname=temperature;location='bedroom';date='saturday';time='7 am') Response: I will turn up the temperature in the bedroom on Saturday at 7 am
Phase 2	Training data	1	Dialog Act: up(functionname=temperature;devicename='airconditioner') Response: I turned up the temperature for the airconditioner
		2	Dialog Act: up(functionname=temperature;devicename='airconditioner';location='bedroom') Response: I turned up the temperature for the airconditioner in the bedroom
		3	Dialog Act: up(functionname=temperature;location='bedroom';date='saturday';time='7 am') Response: I will turn up the temperature in the bedroom on Saturday at 7 am
	Test data	4	Dialog Act: up(functionname=temperature;devicename='airconditioner';time='4 pm';date='saturday';location='living room') Response: I will turn up the temperature of the airconditioner in the living room on Saturday at 4 pm

Table 1: Sample data from IoT dataset. The first line of each example is Dialog Act and the next line is the corresponding response. For test data, the second line is inferred response. Notice that the inferred response is being reused in Phase 2.

Category	Model	BLEU Score (Slot count)			
		1∼2 slots trainset / 3 slots testset	1∼3 slots trainset / 4 slots testset	1∼4 slots trainset / 5 slots testset	1∼5 slots trainset / 6 slots testset
Fully Human Labeled	Seq2Seq w/ attn	70.3	72.3	77.2	82.9
	SC-LSTM	86.6	85.3	80.9	87.4
Self-Learning	Rule-Based	74.2	71.9	72.8	72.9
	Seq2Seq w/ attn	65.3	54.4	45.9	42.9
	SC-LSTM	**86.6**	**85.8**	**84.0**	**94.9**

Table 2: BLEU score Results. Test results for $k = 2$ to $k = 5$

Category	Model	Mean Human Ranking
Fully Human Labeled	Seq2Seq w/ attn	3.16
	SC-LSTM	2.02
Self-Learning	Rule-Based	2.39
	Seq2Seq w/ attn	3.6
	SC-LSTM	**1.55**

Table 3: Human Evaluation Results. For the human ranking, 1 is highest ranked and 5 is lowest ranked.

model as the training set. In the first training iteration, each $\langle DialogAct, Response \rangle$ pair in the dataset consists of a maximum of two slots. For each training iteration, we split the available training dataset into $4 : 1$ partitions for training and validation respectively. For testing, we use human generated $\langle DialogAct, Response \rangle$ pairs which contain one more slot compared to the dataset used in the training step. For comparison, we also trained our neural models using human generated test pairs with less than k slots combined with original train dataset. They are also evaluated using the test dataset which contains $k + 1$ slots per pair.

4.2 Evaluation Metrics

To evaluate our models, we measure BLEU scores and use a human evaluation. The BLEU score is the n-gram similarity between the reference response and the generated response. Higher BLEU scores indicate a better model. The whole test set was used to measure BLEU scores. For human rating, we asked 21 human evaluators to rank the outputs of all five models, considering the **grammatical correctness** and **informativeness** of the generated responses. Then we calculated the Mean Rank for each model. Lower Mean Rank indicates a better model. Randomly chosen 10 responses with 3 to 5 slots were used for human rating.

4.3 Results

The results can be found in Table 2 and Table 3. We were able to train both deep learning models using our self-learning architecture successfully, which can be trained from small amounts of data. The human labeled SC-LSTM model has better scores than the corresponding Seq2Seq model. When self-learning architecture is used, SC-LSTM outperforms all the other models. We think SC-LSTM model is more suitable for learning from structured data than the other models are. This is because SC-LSTM model uses an addi-

tional gated cell which plays the role of sentence planning to produce a surface realization which accurately encodes the input information. When the model encounters a Dialog Act with more slot-value pairs than those in the training dataset, it recognizes the extra slot and generates the response. The Seq2Seq model captures the same relation above using attention. However Seq2Seq-model could not outperform the SC-LSTM model, when it encounters a Dialog Act with more slots than the model was originally trained on. Therefore it fails on our self-learning task. The Rule-based model produces reasonable outputs, but its output heavily depends on the nature of input data and requires human effort. One interesting observation is that the BELU score was the highest when the maximum number of slots was used. We think the BLEU score can increase when the number of target slots rises above a certain level, because the dataset with more slots has more training responses and has responses about more complex combinations of slots.

5 Conclusion

This paper presents a self-learning architecture for NLG. We experimented with three different models: Rule-based, Seq2Seq and SC-LSTM models in the IoT domain. Our data-efficient architecture not only reduces human effort, but also outperforms models trained on the fully human labeled dataset. We think the proposed method can reduce the effort of building large-scale NLG systems for commercial conversational assistant solutions. In future work, we plan to apply other neural models to our architecture, and extend that architecture to cover multi-domain generation tasks.

Acknowledgments

We would like to thank Heriberto Cuayáhuitl and Seonghan Ryu for helpful discussions.

References

Songsak Channarukul, Susan McRoy, and Syed S. Ali. 2002. Jyag & idey: A template-based generator and its authoring tool. In *AAAI/IAAI*.

Adam Cheyer and Didier Guzzoni. 2014. Method and apparatus for building an intelligent automated assistant. US Patent 8,677,377.

Nina Dethlefs. 2017. Domain transfer for deep natural language generation from abstract meaning representations. *IEEE Computational Intelligence Magazine*, 12:18–28.

Ondřej Dušek and Filip Jurčíček. 2016. Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings.

Katja Filippova. 2010. Multi-sentence compression: Finding shortest paths in word graphs. In *COLING*.

Michel Galley, Eric Fosler-Lussier, and Alexandros Potamianos. 2001. Hybrid natural language generation for spoken dialogue systems. In *Proceedings of the 7th European Conference on Speech Communication and Technology (EUROSPEECH01)*, page 17351738, Aalborg, Denmark.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *J. Artif. Int. Res.*, 61(1):65–170.

Albert Gatt and Ehud Reiter. 2009. Simplenlg: A realisation engine for practical applications. In *Proceedings of the 12th European Workshop on Natural Language Generation*, ENLG '09, pages 90–93, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sebastian Gehrmann, Falcon Dai, Henry Elder, and Alexander Rush. 2018. End-to-end content and plan selection for natural language generation.

Kallirroi Georgila, Carla Gordon, Hyungtak Choi, Jill Boberg, Heesik Jeon, and David Traum. 2018. Toward low-cost automated evaluation metrics for internet of things dialogues. In *Proceedings of the 9th International Workshop on Spoken Dialogue Systems Technology (IWSDS)*.

Susan McRoy, Songsak Channarukul, and Syed S. Ali. 2003. An augmented template-based approach to text realization. *Natural Language Engineering*, 9:381–420.

Danilo Mirkovic and Lawrence Cavedon. 2011. Dialogue management using scripts. US Patent 8,041,570.

Karthik Sankaran Narayan, Charles Lee Isbell, and David L. Roberts. 2011. Dextor: Reduced effort authoring for template-based natural language generation. In *AIIDE*.

Kari-Jouko Räihä and Esko Ukkonen. 1981. The shortest common supersequence problem over binary alphabet is np-complete. *Theor. Comput. Sci.*, 16:187–198.

Ehud Reiter. 1996. Building natural-language generation systems. *CoRR*, cmp-lg/9605002.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems.

Li Zhao, Sung Sam Yuan, Sun Peng, and Tok Wang Ling. 2002. A new efficient data cleansing method. In *Proceedings of the 13th International Conference on Database and Expert Systems Applications*, DEXA '02, pages 484–493, London, UK, UK. Springer-Verlag.