RAD: Reinforced Attention Decoder Model On Question Generation

Xin Li, Zhen Huang, Feng Liu*, Changjian Wang, Minghao Hu, Shiyi Xu, Yuxing Peng
Science and Technology on Parallel and Distributed Laboratory, National University of Defense Technology
Changsha, China
{lx, huangzhen, richardlf, wangchangjian, xushiyi18}@nudt.edu.cn; huminghao16@gmail.com; pengyuxing@aliyun.com

***Abstract*—Question Generation (QG) aims to construct questions from given text automatically. Recently, QG has received widely concerned. The mainstream method is still based on the fixed sequence generation model of Seq2Seq model, and few people consider the influence of generation order in the result. In this paper, we present a novel Reinforced Attention Decoder Neural Network for the QA-SRL task. First, our model draws on the idea of the reinforcement learning algorithm with baseline, which is using the accuracy of each slot and sentence as the reward, updating the Policy Network to predict the optimal generation order. Second, we apply the Attention mechanism on the baseline to get more relevant information about the entire sentence. Addition experiments explore that distilling knowledge from RAD (the teacher) model to guide RAD-Reborn model (the student) training can achieve better performance. Extensive experiments on QA-SRL Bank 2.0 show that our model outperforms previous systems of all of the evaluation metrics. In particular, the metric EM (Exact Match) increased significantly by over 3%.**

***Index Terms*—question generation, reinforcement learning, semantic role labeling, reborn network, encoder-decoder**

I. Introduction

Question generation aims to construct natural questions from a given sentence or paragraph automatically. With the evolution of large-scale QA datasets in recent years, current QA models can even outperform humans [1]. For example on SQuAD [2], a common QA benchmark dataset, BERT-based models [3] even beat human's performance; TriviaQA [4], a large-scale QA dataset, on which human's performance has been reached. In all these cases, very large amounts of training data are available. However, a basic task to construct such large-scale dataset is to generate natural questions.

Generally, there are following ways to generate questions in datasets:

- **Crowdsource annotation:** To construct SQuAD [2], the annotators need to read the context and then annotate the question corresponding to the entity in the text individually. However, they suffer from some problems, such as low generation speed, high cost and great difference in the process.
- **Rule-based generator:** The success of these approaches depends on the existence of well-designed rules for declarative-to-interrogative sentence transformation, typically based on deep linguistic knowledge. However, such methods have been proven empirically weak for QA [5].
- **Neural-based generator:** The neural-based approaches are typically end-to-end models which generate question automatically by Seq2Seq [6] mode. The attention mechanism [7] is applied to help decoder pay attention to the most relevant parts of the input sentence while generating a question. The neural model then uses the attention mechanism as a default [8] [9] [10]. Besides, almost all neural generation models' decoder commonly assumes a left-to-right generation order. Therefore, in the generation process, the first few tokens are definitely more accurate than the latter [11]. To alleviate the problem, several researchers [11] [12] [13] use the synchronous and asynchronous bidirectional decoding Seq2Seq model in machine translation to improve performance. In fact, because of the imbalanced distribution of data, some slots are difficult to predict. If we put such slots in the back of prediction sequence, it will further worsen the result. Gu et al. (2019) [14] shows that changing the order of training has an effective on the generation.

TABLE I
AN EXAMPLE OF QA-SRL

Lee **went** home happily in the afternoon

answer	***question***
Lee	Who went somewhere?
home	Where did someone go?
happily	How did someone go somewhere?
in the afternoon	When did someone go somewhere?

In this paper, we focus on two important issues for question generation: (1) How to find a relatively good order to generate questions to reduce the impact of error accumulation? (2) How to make full use of the information of the whole sentence in the generation process?

To overcome the challenges, we proposed a **R**einforced **A**ttention **D**ecoder Model On Question Generation, which is referred to as RAD. Then we carry out several experiments on two subsets of QA-SRL, which is a large-scale QA [15] dataset constructed by model. The QG baseline model utilized the detected argument span for a predicate to generate questions

*Corresponding author.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

to label the semantic relationship in QA-SRL [15]. As shown in TABLE I, the question starts with wh-words (Who, What, Where, What, How, etc.). Each sentence contains at least one verb predicate, and the answers are phrases in the sentence. Evaluations show that our model achieves 52.0%, 68.2% and 85.1% on Exact Match (EM), Partial Match (PM) and Slot-level accuracy (SA), all of which are better than the strong baseline in [15].

Our main contributions are listed as follows:

- We propose a reinforced reordering policy to learn the optimum sequence generation order. We sample the action with the highest score as the output and use the distributed sampling order as the baseline, take the difference between the two as the total reward, and update the Policy Network to predict the optimal generation order.
- Moreover, we utilize Born Again neural network [16] (RAD-Reborn) that distill knowledge from teacher network to retrain the student network for further improve the performance.
- We integrate attention mechanism in the decoding stage, make the model focus on finding useful information more significantly related to the current output in the input context, combine the information of the sentence, the answer and the generated word, to improve the quality of question generation.

This paper is organized as follows. We discuss related work in section II, describe the details of our models in Section III, especially on the Attention and Reinforced Reordering Policy Mechanism. Then, we present the evaluation results as well as a detailed analysis in Section IV. Finally, we conclude the paper in Section V.

II. Related Work

A. Question Generation Method

Recently, driven by advances in deep learning, "neural" techniques have become the mainstream. Du et al. (2017) [5] pioneered the first NQG model using an attention Seq2Seq model [7], which fed a sentence into a RNN-based encoder, and generated a question about the sentence through a decoder. Some researchers treated the answer's position as an extra input feature [17] [18] or encoded the answer with a separate RNN [9] [19]. A few works [9] [20] considered question word generation separately in model design. Zhao et al. (2018) [18] proposed a gated self-attention encoder to refine the encoded context by fusing important information with the context's self-representation properly, which has achieved state-of-the-art result on SQuAD. Relevant QG work [21] [22] have adopted policy gradient methods to add task specific rewards (such as BLEU or ROUGE) to the original objective. This helped to diversify the questions generated, as the model learned to distribute probability mass among equivalent expressions rather than the single ground truth question.

B. Sequence Order

Seq2Seq models suffer from a weakness, which is the problem of unbalanced outputs (the prefixes are better predicted than the suffixes). It is mainly caused by the accumulation of errors [11]. Recently, several researchers have proposed to adjusting the order of prediction to alleviate the problem of error accumulation. Zhang et al. (2018) [23] presented an asynchronous bidirectional decoding algorithm for NMT, which extended the conventional encoder-decoder framework by utilizing a backward decoder. Zhou et al. (2019) [13] proposed a novel framework (SB-NMT) that utilized a single decoder to generate target sentences bidirectionally, simultaneously and interactively. Gu et al. (2019) [14] proposed to train the generated sequence according to the self-adaptive sequence of the model.

Differently, our method aims to find an optimal order for prediction using reinforcement learning.

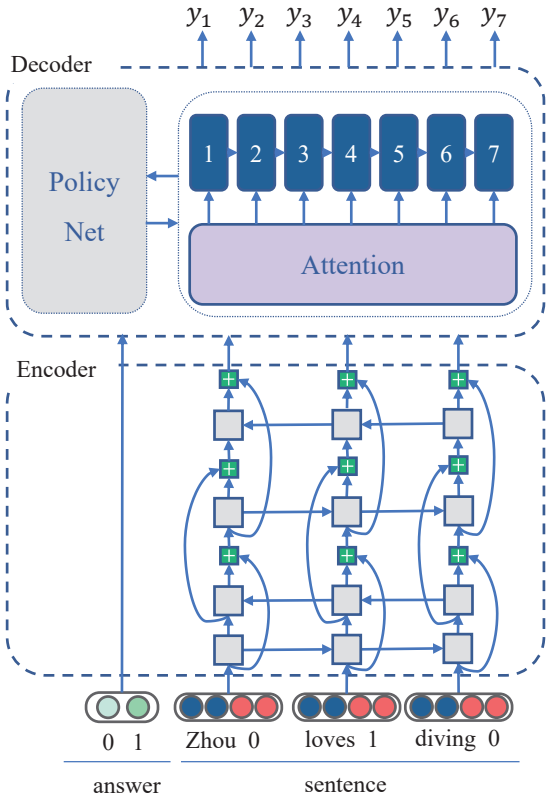


Fig. 1. Overall model

III. Model

Formally, we define the task as follows. Given a sentence $X = \{x_1, x_2, \cdots, x_n\}$ and the answers $s_{ij} \in \{(i,j)|i,j \in [0,n], j \geqslant i\}$, where S is the set of s_{ij}, the Question Generation model aims to generate a set of questions $Q = \{q_1, q_2, \ldots, q_m\}$ for each span in S, where m is the size of S. Specifically, the q_i has a fixed syntactic structure of seven slots, each with a separate vocabulary. Then the model can be designed to predict the joint distribution $p(\boldsymbol{y}|X, s_{ij})$ over words $\boldsymbol{y} = (y_1, \ldots, y_7)$ for question slots, which can be

trained by minimizing the negative log-likelihood of golden slot values. As shown in Fig.1, a RNN encoder-decoder architecture [7] is utilized, where we introduce the details of encoder in section III-A and decoder in section III-B. Specifically, we detailed illustrate (1) the global attention mechanism [24] to make the model focus on certain elements of the input in subsection III-B1, (2) the reinforced reordering policy to find an optimal order for prediction in subsection III-B2 and (3) reborn training mechanism to train a better model in subsection III-C.

A. Encoder

The encoding method is implemented as [25] proposed. We start with an input $X = \{x_1, \ldots, x_n\}$, where the representation x_i at each time step is a concatenation of the token embedding w_i and an embedded binary feature, which indicates whether x_i is the predicate or not. We then compute the output representation $H = BILSTM(X)$ using a stacked alternating LSTM [26] with highway connections [27] and recurrent dropout [28]. And the answer is the concatenation of the hidden states of $BILSTM$ at each endpoint. For example, the answer representation Ans_{ij} of s_{ij} is

$$Ans_{ij} = [H[i], H[j]] \tag{1}$$

B. Decoder

1) Attention: In order to take full advantage of the encoded information of the sentence when generating tokens at each step, we use the global attention mechanism [7] to focus on the overall information of the sentence, not only the answer [15]. As shown in Fig.2, we at first use the match function to calculate the matching score e_{it} of the encoding vector of each token h_i and decoder's hidden state at the previous step h_{t-1}, and then use the softmax function to normalize the score e_{it} to an attention value α_{it}. Taking α_{it} as the weight, we sum the weighted output of encoder H and calculate the attention vector att_t at step t. Next, the decoder uses the output of the previous RNN unit h_{t-1}, the answer representation Ans_{ij}, and the attention vector att_t to get the output at step t h_t as follows.

At each slot t, we apply L layers of LSTM cells to obtain h_t:

$$h_t, c_t = LSTM^L(h_{t-1}, c_{t-1}) \tag{2}$$

where the c_t is the cell state that holds the long term state.

Specifically, we use the span representation Ans_{ij}, the output of the previous LSTM h_{t-1} and the attention vector att_{t-1} to update LSTM unit:

$$hidden_{t-1} = Linear(h_{t-1}) \tag{3}$$

$$h_{t-1} = [hidden_{t-1} : Ans_{ij} : att_{t-1}] \tag{4}$$

where $hidden_{t-1}$ is the vector obtained by dimensional transfer function $Linear()$. [:] is concatenation operation.

Here the global attention mechanism is calculated by the Eq.5 at step t:

$$att_t = \textstyle\sum_{i=1}^{|X|} \alpha_{it} H_i \tag{5}$$

$$\alpha_{it} = \frac{exp(e_{it})}{\sum_{j=1}^{|X|} exp(e_{jt})} \tag{6}$$

$$e_{it} = match(H_i, h_{t-1}) \tag{7}$$

where H_i is the i-th output of encoder, α_{it} is the attention weight of step t with H_i; e_{it} is the match value using the $match()$ function, here we use the dot product to calculate match value. The initial state of Decoder h_0 is the average sum of all H.

2) Reinforced Reordering Policy: Many current NQG models follow the Seq2Seq architecture. The usual decoding process is to generate tokens one by one according to the order of sentences. Decoding step can be defined as finding the question $\mathbf{y}$ that maximizes the conditional likelihood given the passage X and the answer S:

$$\bar{\mathbf{y}} = \underset{\mathbf{y}}{argmax}\, P(\mathbf{y}|X, S) \tag{8}$$

$$= \underset{\mathbf{y}}{argmax} \sum_{t=1}^{m} P(y_t|X, S, y < t) \tag{9}$$

where m is the length of generated questions.

Because of error propagation, the prediction of suffixes are worse than the prefixes. If we put the slots of poor prediction results behind the prediction sequence, it will further lead to the deterioration of the prediction results. Here, in order to alleviate the problem above, we propose Reinforced Reordering Policy (RRP in short) mechanism to find an optimal order for prediction. As shown in Fig.3, we model the Policy Network using a fully connected network. And the sentence accuracy EM and each slot accuracy SA are taken as reward to update the Policy Network. Through multiple sampling, the order's distribution $p(A|\theta)$ converges to obtain an optimal generation order to generate questions, to reduce the influence of explosion bias during training. So, the model aims to find the order $A = [a_1, a_2, \ldots, a_m]$ to generate questions:

$$\bar{\mathbf{y}} = \underset{\mathbf{y}}{argmax} \textstyle\sum_{t=1}^{m} P(y_{A_t}|X, S, y < t) \tag{10}$$

Specifically, we use the policy gradient REINFORCE with baseline algorithm [29] to update the Policy Network. Here, we define the reward $R(A)$ when the order is A:

$$R(A) = EM_A + \textstyle\sum_{i=1}^{m} \gamma^{m-i} SA_{A_i} \tag{11}$$

Here γ is the loss hyper-parameter. EM_A denotes the sentence accuracy in order A, SA_{A_i} is the accuracy of $i-th$ slot in order A. The function is designed to encourage low accuracy slots to be placed in the initial prediction to reduce the impact of explosion bias on them. The training objective $J(\theta)$ is to minimize the negative expected reward by

$$J(\theta) = -E_{A^a \backsim p_\theta(A)}[R(A^a) - R(A^s)] \tag{12}$$

$$A^a = argmax(p(A|\theta)) \tag{13}$$

$$A^s = Sample(p(A|\theta)) \tag{14}$$

where we abbreviate the policy distribution $p(A|\theta)$ as $p_\theta(A)$. A^a is the order sampled using argmax function, and A^s is

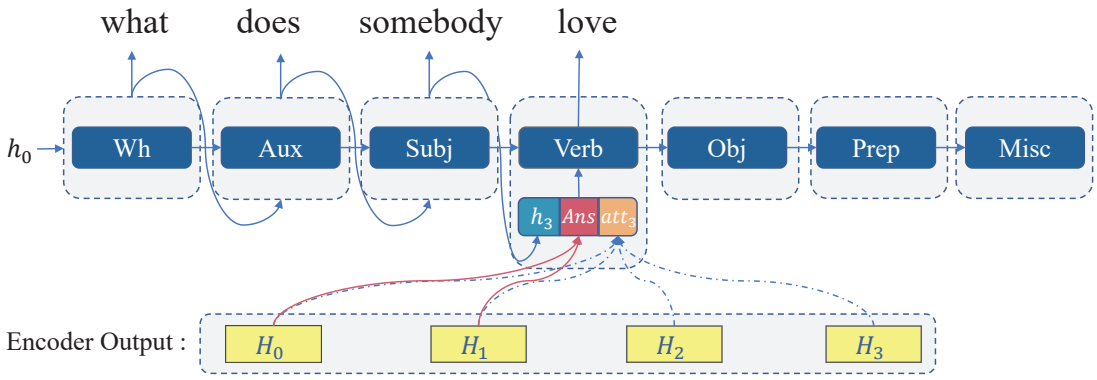


Fig. 2. Attention Mechanism

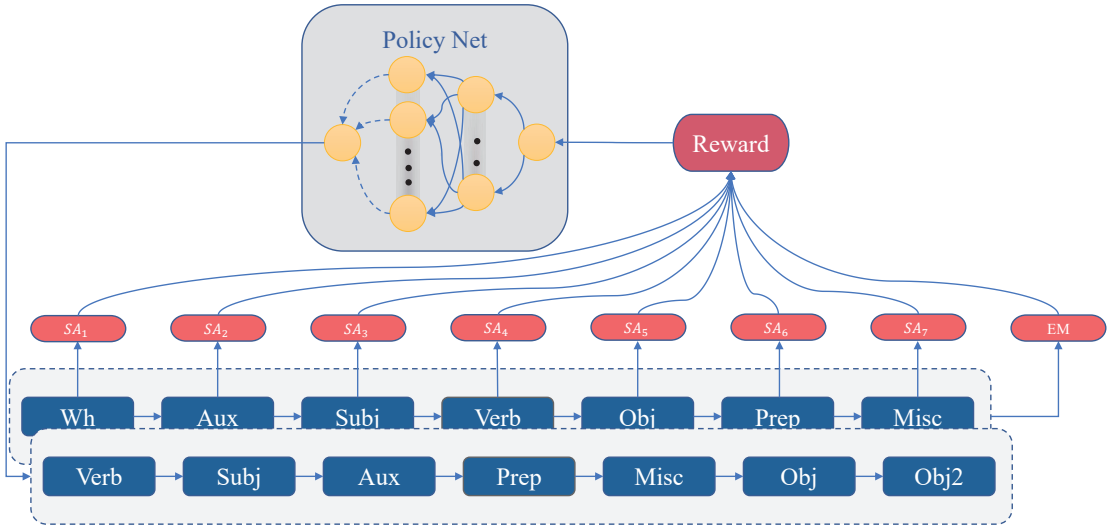


Fig. 3. Reinforced Reordering Policy

obtained by sampling according to the distribution $p(A|\theta)$. We use Eq.11 to calculate their reward $R(A^a)$ and $R(A^s)$. Such approach is known as the self-critical sequence training (SCST) [30], which is first used in image caption. It can be used to normalize the reward and reduce variances [31].

According to the REINFORCE algorithm [29], the gradient $\nabla J(\theta)$ can be computed as:

$$\begin{aligned}\nabla_\theta J(\theta) &= -E_{A^a \backsim p_\theta(A)}[(R(A^a) - b)\nabla_\theta log p_\theta(A^a)] \\ &\approx -(R(A^a) - R(A^s))\nabla_\theta log p_\theta(A^a)\end{aligned} \tag{15}$$

where b is baseline to reduce variances. Here, we take the sampled A^s as b. In this case, this action is encouraged if A^a is superior to the A^s.

The output distribution at slot t is computed via the final output vector h_t:

$$p(y_t|X, s_{ij}) \propto exp(w_k^T MLP(h_t) + d_t) \tag{16}$$

where the d_t is bias at step t.

C. Reborn Training

Reinforcement learning is to obtain a good strategy by constantly exploring and updating the strategy network with feedback information. During the experiment, we found that the convergence curve of the model would jitter under the influence of multiple samples, thus affecting the performance of the model, the detail in subsection IV-G. Combined with knowledge distillation [32], Furlanello et al. (2018) [16] proposed Born-Again neural networks: transferring "knowledge" from one machine learning model (the teacher) to another (the student), we train students parameterized identically to their teachers. Experiments show that the performance of the Born-Again network (student) exceeds that of the teacher network.

Inspired by [16], we propose the Reborn Training Mechanism: after the teacher network convergence, we re-initialize a new student network (a student network with the same number of parameters as the teacher network), retrain the student model by using the convergence results of teacher model. As shown in Fig.4, we use the order which is converged by teacher

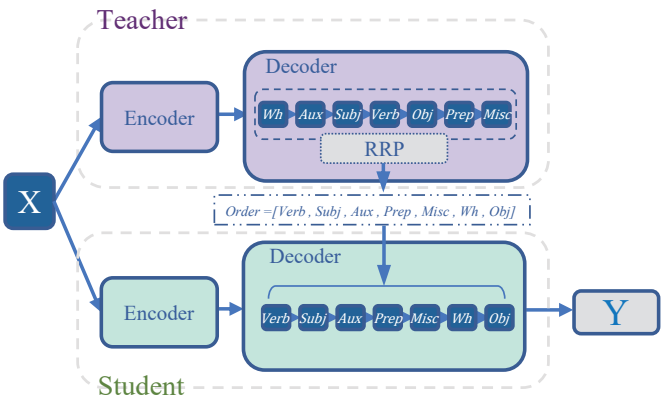


Fig. 4. Reborn Training Mechanism.

model (RAD) to train the student network (without RRP), so as to reduce the performance loss due to sampling.

IV. Experiment

In this section, we illustrate the dataset in subsection IV-A, the implementation detail in subsection IV-B, the evaluation metrics in subsection IV-C, the overall results in subsection IV-D, the ablation studies in subsection IV-E. In addition, we analysis the effectiveness of Reinforced Reordering Policy and the Reborn Training Mechanism in subsection IV-F and IV-G. Finally, we make the error analysis in subsection IV-H.

A. Dataset

We focus on the QA-SRL dataset [15] to train and evaluate our model. The corpus we experimented, QA-SRL Bank 2.0, consists of over 250,000 question-answer pairs for over 64,000 sentences across 3 domains and is gathered with a new crowd-sourcing scheme that we show has high precision and good recall at modest cost. In addition, we also test our model on the higher-density version dataset. As shown in table II, we list the details of the train, dev and test sets. There are 2 QA pairs on average annotated for each sentence in Orig set, and 4.8 QA pairs in Dense set.

TABLE II
Statistics for Datasets.

	Train	Dev		Test	
		Orig	Dense	Orig	Dense
Instances	90,265	16,601	5,837	19,445	5,768
Verbs	95,285	17,581	5,886	20,613	5,844
QA pairs	189,554	34,053	28,161	39,887	26,043

B. Implementation Detail

We evaluate the RAD model by running the following setting. We use the Adam optimizer for training [33]. The initial learning rate is 0.01, and halved whenever meeting a bad iteration. The batch size is 80, and a dropout rate of 0.3 is used to prevent overfitting. The model uses ELMo deep contextualized word representations. The encoder hidden size is 300, the decoder hidden size is 200, and the loss hyper-parameter γ is 0.7.

C. Evaluation Metrics

Three evaluation metrics [15] are used: (1) Exact Match accuracy (EM). EM denotes the accuracy at which the predicted question exactly matches the golden question; (2) Partial Match accuracy (PM). PM is a relaxed match that considers only question word (WH), subject (SBJ), object (OBJ) and Miscellaneous (Misc) slots; (3) Slot-level accuracy (SA). SA is the average accuracy of all question slots. Among them, EM is the main metric. Since EM is the probability that all slots match, approximately the multiplications of the accuracy of all slots, it is difficult to improve.

D. Overall Result

All models are trained on the origin training set of QA-SRL, and evaluated on the origin and dense development set. Here, we use the model in [15] as the baseline for comparing results. Baseline uses a 4-layer LSTM to encode a sentence, and a 4-layer LSTM to generate questions. As shown in Table III, all evaluation metrics of RAD are superior to baseline and RAD-Reborn achieves the best performance. As we can see, the EM value is significantly improved by more than 3% in both two development sets, indicating that the order RAD learned can improve the decoding performance. In particular, our model can improve Sentence Accuracy (EM) effectively. The RAD-Reborn model, which is retrained on the order that learned from RAD, achieves the best performance. It demonstrates that Reborn Training is effective.

TABLE III
Performance comparison of baseline, RAD and RAD-Reborn on the development sets.

Model	Orig			Dense		
	EM	*PM*	*SA*	*EM*	*PM*	*SA*
Baseline [15]	48.1%	66.7%	83.9%	41.6%	59.1%	81.9%
RAD	51.3%	67.4%	84.8%	42.9%	59.0%	82.0%
RAD-Reborn	**52.0%**	**68.2%**	**85.1%**	**44.1%**	**59.7%**	**82.5%**

E. Ablation Studies

The contributions of each component of our model are shown in Table IV.

1) Attention Mechanism: As can be seen from Ablation (1), all the metrics get worse without Attention Mechanism. In comparison with (2), (3), removing Attention also lead to the performance degradation in the same case. It indicates that the Attention Mechanism is effective in this task.

In addition, we explored the effect of decoding layer L on model performance. As shown in Fig.5, we notice that as the number of layers increases, the performance of the model decreases. We argue that increasing the number of layers in the decoding process will cause the attenuation of the decoded information. One layer is the best choice.

TABLE IV
ABLATION STUDY ON ORIG DEV SET.

Test on Orig	**EM**	**PM**	**SA**	**ΔEM**	**ΔPM**	**ΔSA**
(0) **RAD(ours)**	**51.3**	**67.8**	**84.8**	-	-	-
(1) - Attention	51.0	67.6	84.8	-0.3	-0.2	-
(2) - RRP	49.9	67.7	84.7	-1.4	-0.1	-0.1
(3) - RRP - Attention	49.1	66.5	83.9	-2.2	-0.9	-0.9

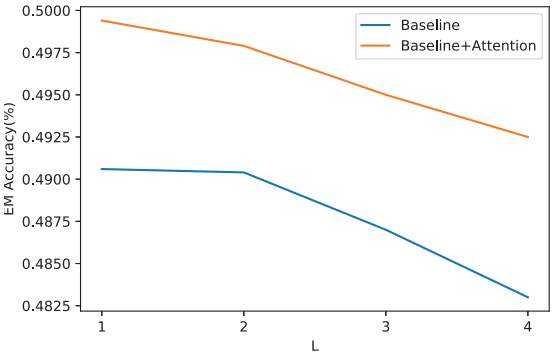


Fig. 5. Effect of number of decoding layers on performance.

2) Reinforced Reordering Policy: Ablation (2) shows the effectiveness of RRP. We notice that the EM value drops by 1.4% when we remove RRP, and the other metrics have not changed much. It proves that adjusting the prediction order has a greater impact on the EM value. It shows that training from left to right is not the best option. Through dynamic learning with our approach, a superior order can be obtained to train a better model.

F. Effectiveness of RRP

We do further experiment to demonstrate the effectiveness of RAD. We list the order of the models' final predictions (Baseline and RAD), as shown in table V. Meanwhile, we obtain the accuracy of each slot when it converges to the optimal order, as shown in table VI.

TABLE V
THE ORDER MODELS LEARNED.

Model	**Order**
Baseline [15]	'wh', 'aux', 'subj', 'verb', 'obj', 'prep', 'obj2'
RAD	'verb', 'wh', 'obj2', 'aux', 'subj', 'obj', 'prep'

As can be seen from table VI, except for "wh" and "obj", the accuracy of all other slots has been improved, among which the accuracy of "verb" has been increased by 4.2%. Combining table V and table VI and considering the characteristics of Seq2Seq model, we wonder whether the model would get better results by putting the less accurate slots in front. Is it because the "wh" and "obj" positions moved back that the performance did not improve? To verify the conjecture, we design a Heuristic Algorithm: in the validation step after one

TABLE VI
THE SLOT ACCURACY OF EACH SLOT.

	Baseline [15](%)	**RAD(%)**	**△(%)**
wh	91.1	90.8	-0.3
aux	74.0	74.5	+0.5
subj	88.1	88.2	+0.1
verb	69.9	74.1	+4.2
obj	90.8	90.8	-
prep	86.8	87.2	+0.4
obj2	86.8	87.9	+1.1

epoch, we adjust the training order: put the slots with low accuracy in front of them for training manually, and after several adjustments, the model converges to the final order. The order is: $['verb','aux','prep','obj2','subj','obj','wh']$ The experimental results are shown in table VII, the result of Heuristic Algorithm is indeed better than the baseline model, but not as good as the result obtained by RAD. We can see that the order which RAD produced is not always from low to high accuracy. By comparing the vocabulary of every slot, we found that the vocabulary size of each slot was different, which would make the prediction difficulty varies. So, we use the accuracy of each slot and sentence accuracy to learn an order dynamically is a better way.

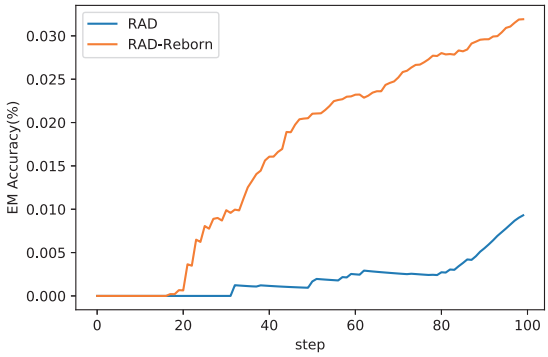
TABLE VII
PERFORMANCE COMPARISON OF OUR MODELS ON THE ORIGIN DEV SET.

Test on Orig	**EM**	**PM**	**SA**
Baseline [15]	48.1	66.7	83.9
Heuristic Algorithm(ours)	51.0	67.3	84.7
RAD(ours)	51.3	67.4	84.8
RAD+Reborn(ours)	**52.0**	**68.2**	**85.1**

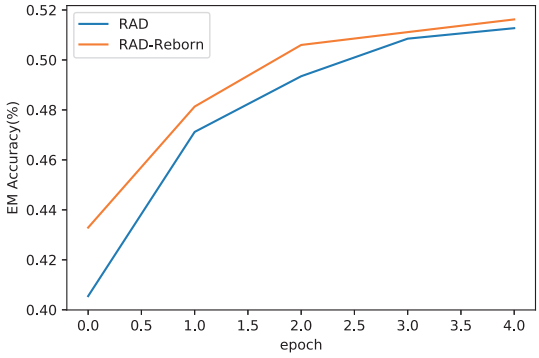
G. The Reborn

As we can see from table VII, the RAD-Reborn achieves the best performance. In order to explore the reasons, we use the best order of learning from the teacher network (RAD) to train the student network (RAD-Reborn). (a) in Fig.6 shows the curve of EM value as one of the rewards changing with sampling steps at the beginning of the training of the two models. (b) is the comparison curve of the EM validation result of the two models after each epoch. From (a), we can see that the RAD convergence curve is jittering during multiple sampling, and the model does not begin to converge rapidly until the Policy Net samples 80-100 times. This indicates that the sampling process causes the model to fail to converge to the best performance, and adjusting the

training mode of the network model constantly is detrimental to performance. RAD-Reborn achieves best performance so far. It proves that Reborn Training Mechanism, which use the knowledge distillation from teacher network (RAD) to train student network (RAD-Reborn), is effective.



(a) Training



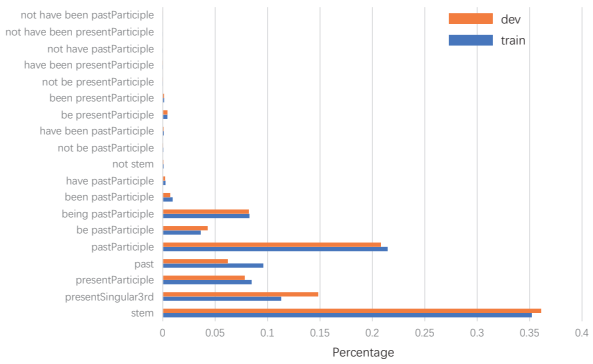
(b) Validating

Fig. 6. The EM curves during training and validating.

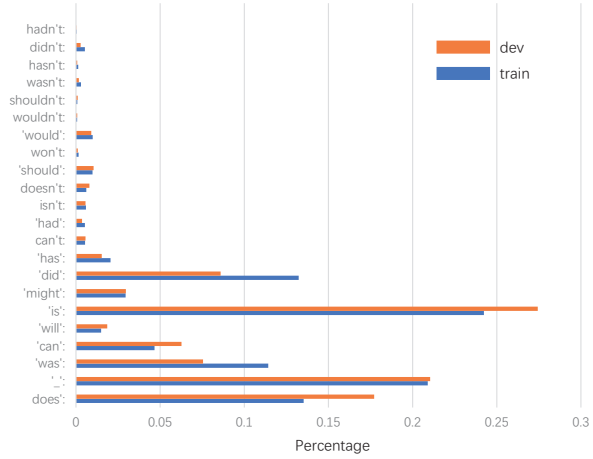
H. Error Analysis

In table VI, we notice that 5 slots (wh, subj, obj, prep, obj2) for accuracy above 86%, but 2 slots (aux, verb) below 74%. When we use Heuristic Algorithm (putting 'verb' and 'aux' in front to train), the improvements of 'verb' and 'aux' are still not significant, the verb's generation performance is the worst. In addition to the model itself, we attribute this to differences in the difficulty of generating each slot token due to the characteristics of the datasets.

Fig.7 shows the distribution of 'verb' and 'aux' in train and dev sets. The first seven types of 'verb' make up more than 98% of all kinds, and the first nine of 'aux' occupy more than 94%. We argue that the wide variety and unbalance distribution of dataset result in their poor performance. In addition, since the model does not consider the tense of slot 'verb' during encoding, it is more difficult to predict 'verb' slot.



(a) 'verb'



(b) 'aux'

Fig. 7. The distribution of 'verb' and 'aux' in train and dev sets.

V. Conclusion

We propose the Reinforced Attention Decoder, a generation model combining reinforcement learning algorithm and attention mechanism. We have three main contributions. First, we improve the error accumulation problem of Seq2Seq model by proposing a reinforced reordering policy to learn the optimum sequence generation order. Second, we propose the Reborn Training Mechanism that distill knowledge from teacher network to retrain the student network for further improve the performance. Third, we integrate attention mechanism in the decoding stage, make the model focus on finding useful information more significantly related to the current output in the input context. Our model combines the information of the sentence, the answer and the generated word to improve the performance. Our model outperforms the existing approaches by more than 3% on QA-SRL dataset. We make the error analysis and find that the tense information was helpful for verb prediction. We will work on this issue in the future.

References

[1] Z. Huang, S. Xu, M. Hu, X. Wang, J. Qiu, Y. Fu, Y. Zhao, Y. Peng, and C. Wang, "Recent trends in deep learning based open-domain textual question answering systems," *IEEE Access*, 2020.

[2] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," *arXiv preprint arXiv:1606.05250*, 2016.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[4] M. Joshi, E. Choi, D. S. Weld, and L. Zettlemoyer, "Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension," *arXiv preprint arXiv:1705.03551*, 2017.

[5] X. Du, J. Shao, and C. Cardie, "Learning to ask: Neural question generation for reading comprehension," *arXiv preprint arXiv:1705.00106*, 2017.

[6] I. Sutskever, O. Vinyals, and Q. Le, "Sequence to sequence learning with neural networks," *Advances in NIPS*, 2014.

[7] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[8] Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," *arXiv preprint arXiv:1703.03130*, 2017.

[9] N. Duan, D. Tang, P. Chen, and M. Zhou, "Question generation for question answering," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 866–874.

[10] V. Harrison and M. Walker, "Neural generation of diverse questions using answer focus, contextual and linguistic features," *arXiv preprint arXiv:1809.02637*, 2018.

[11] L. Liu, M. Utiyama, A. Finch, and E. Sumita, "Agreement on target-bidirectional neural machine translation," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 411–416.

[12] R. Sennrich, B. Haddow, and A. Birch, "Edinburgh neural machine translation systems for wmt 16," *arXiv preprint arXiv:1606.02891*, 2016.

[13] L. Zhou, J. Zhang, and C. Zong, "Synchronous bidirectional neural machine translation," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 91–105, 2019.

[14] J. Gu, Q. Liu, and K. Cho, "Insertion-based decoding with automatically inferred generation order," *arXiv preprint arXiv:1902.01370*, 2019.

[15] N. FitzGerald, J. Michael, L. He, and L. Zettlemoyer, "Large-scale qa-srl parsing," *arXiv preprint arXiv:1805.05377*, 2018.

[16] T. Furlanello, Z. C. Lipton, M. Tschannen, L. Itti, and A. Anandkumar, "Born again neural networks," *arXiv preprint arXiv:1805.04770*, 2018.

[17] Q. Zhou, N. Yang, F. Wei, C. Tan, H. Bao, and M. Zhou, "Neural question generation from text: A preliminary study," in *National CCF Conference on Natural Language Processing and Chinese Computing*. Springer, 2017, pp. 662–671.

[18] Y. Zhao, X. Ni, Y. Ding, and Q. Ke, "Paragraph-level neural question generation with maxout pointer and gated self-attention networks," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 3901–3910.

[19] Y. Kim, H. Lee, J. Shin, and K. Jung, "Improving neural question generation using answer separation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 6602–6609.

[20] X. Sun, J. Liu, Y. Lyu, W. He, Y. Ma, and S. Wang, "Answer-focused and position-aware neural question generation," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 3930–3939.

[21] X. Yuan, T. Wang, C. Gulcehre, A. Sordoni, P. Bachman, S. Subramanian, S. Zhang, and A. Trischler, "Machine comprehension by text-to-text neural question generation," *arXiv preprint arXiv:1705.02012*, 2017.

[22] V. Kumar, K. Boorla, Y. Meena, G. Ramakrishnan, and Y.-F. Li, "Automating reading comprehension by generating question and answer pairs," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2018, pp. 335–348.

[23] X. Zhang, J. Su, Y. Qin, Y. Liu, R. Ji, and H. Wang, "Asynchronous bidirectional decoding for neural machine translation," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[24] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[25] L. He, K. Lee, M. Lewis, and L. Zettlemoyer, "Deep semantic role labeling: What works and what's next," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 473–483.

[26] J. Zhou and W. Xu, "End-to-end learning of semantic role labeling using recurrent neural networks," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2015, pp. 1127–1137.

[27] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Advances in neural information processing systems*, 2015, pp. 2377–2385.

[28] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *Advances in neural information processing systems*, 2016, pp. 1019–1027.

[29] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[30] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel, "Self-critical sequence training for image captioning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7008–7024.

[31] M. Hu, Y. Peng, Z. Huang, X. Qiu, F. Wei, and M. Zhou, "Reinforced mnemonic reader for machine reading comprehension," *arXiv preprint arXiv:1705.02798*, 2017.

[32] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[33] M. Tang, L. Qiao, Z. Huang, X. Liu, Y. Peng, and X. Liu, "Accelerating sgd using flexible variance reduction on large-scale datasets," *Neural Computing and Applications*, pp. 1–12, 2019.