Separate Answer Decoding for Multi-class Question Generation

Kaili Wu, Yu Hong, Mengmeng Zhu, Hongxuan Tang, Min Zhang
School of Computer Science and Technology, Soochow University
Suzhou, China
{*klwu,mmzhu,hxtang*}*@stu.suda.edu.cn,tianxianer@gmail.com,minzhang@suda.edu.cn*

***Abstract*—Question Generation (QG) aims to automatically generate questions by understanding the semantics of source sentences and target answers. Learning to generate diverse questions for one source sentence with different target answers is important for the QG task. Despite of the success of existing state-of-the-art approaches, they are designed to merely generate a unique question for a source sentence. The diversity of answers fail to be considered in the research activities. In this paper, we present a novel QG model. It is designed to generate different questions toward a source sentence on the condition that different answers are regarded as the targets. Pointer-Generator Network(PGN) is used as the basic architecture. On the basis, a separate answer encoder is integrated into PGN to regulate the question generating process, which enables the generator to be sensitive to attentive target answers. To ease the reading, we name our model as APGN for short in the following sections of the paper. Experimental results show that APGN outperforms the state-of-the-art on SQuAD split-1 dataset. Besides, it is also proven that our model effectively improves the accuracy of question word prediction, which leads to the generation of appropriate questions.**

***Keywords*-Question Generation; Target Answer; Question Word;**

I. Introduction

Question Generation (QG) is an important task in Question Answering (QA). Rus et al. [1] gave the definition of the QG in 2010, that is, given a free text, automatically generate a natural question according to the text. Since then, relevant researches have been conducted and applied to medical and educational fields [2], [3]. In addition, the QG task can generate a large number of question-answer pairs, which can be used to expand the corpus of the QA task and assist in the construction of the QA system.

Previous work for the QG task mainly utilized manually written rules to transform a sentence into a question [4], [2], [5]. However, the method is usually applied to a specific field and difficult to process large-scale data. Compared to the rule-based method, neural network-based method [6], [7], [8] rely on no rules, and it is in an end-to-end fashion driven by large-scale data. Even though, the shortcoming of this method is there are still gaps with natural questions in the expression.

In this paper, we focus on sentence-level QG task which takes a declarative sentence and a target answer as inputs and generates a related question. By anaylzing the datasets, we find a phenomenon a source sentence may contain several target answers which are corresponding to different questions. As shown in Example 1, there are two target answers underlined in the source sentence. Our baseline model PGN [9] can only generate the same question ("*PGN Q1*", "*PGN Q2*") for the source sentence ("*S*"). As we can see that "*Jerome Green*" is the answer to this question, but "*Around 1899*" cannot be used to answer the question. To overcome the limitations, we fuse target answers based on PGN. Concretely, we encode separately the target answer and initialize the decoder state. "*APGN Q1*" and "*APGN Q2*" are generated questions of our model. Those questions are close to the standard questions "*Q1*" and "*Q2*".

Example 1:
***S**: Around 1899, professor Jerome Green became the first American to send a wireless message.*
***Q1**: In what year did Jerome Green send his first wireless message?*
***PGN Q1**: Who became the first American to send a wireless message?*
***APGN Q1**: When did Jerome Green become the first American to send a wireless message?*
***Q2**: Which professor sent the first wireless message in the USA?*
***PGN Q1**: Who became the first American to send a wireless message?*
***APGN Q2**: Who became the first American to send a wireless message?*

We propose a simple and effective method for improving QG performance. Moreover, we alleviate the problem in which a source sentence contains several target answers. Furtherly, we find our proposed method help to generate correct question words. These question words will generate subsequence words in the right way.

II. Related Work

QG task has been mainly tackled with two types of methods: rule-based and neural network-based. Rule-based QG depends on deep linguistic knowledge and well-designed rules for transforming declarative sentences to questions. Lindberg et al. [4] proposed a complex template-based method combined with semantic tagging information. Heilman et al. [2] overgenerated questions by using manually written rules. Then these questions were ranked by a logistic regression model. In addition, Liu et al. [5] used lexical and syntactic information to generate questions in Chinese. Rule-based methods have low universality and rely heavily on rules. Generally, rule-based methods pay more attention to the syntactic roles of words, but not their semantic roles.

978-1-7281-5014-7/19/$31.00

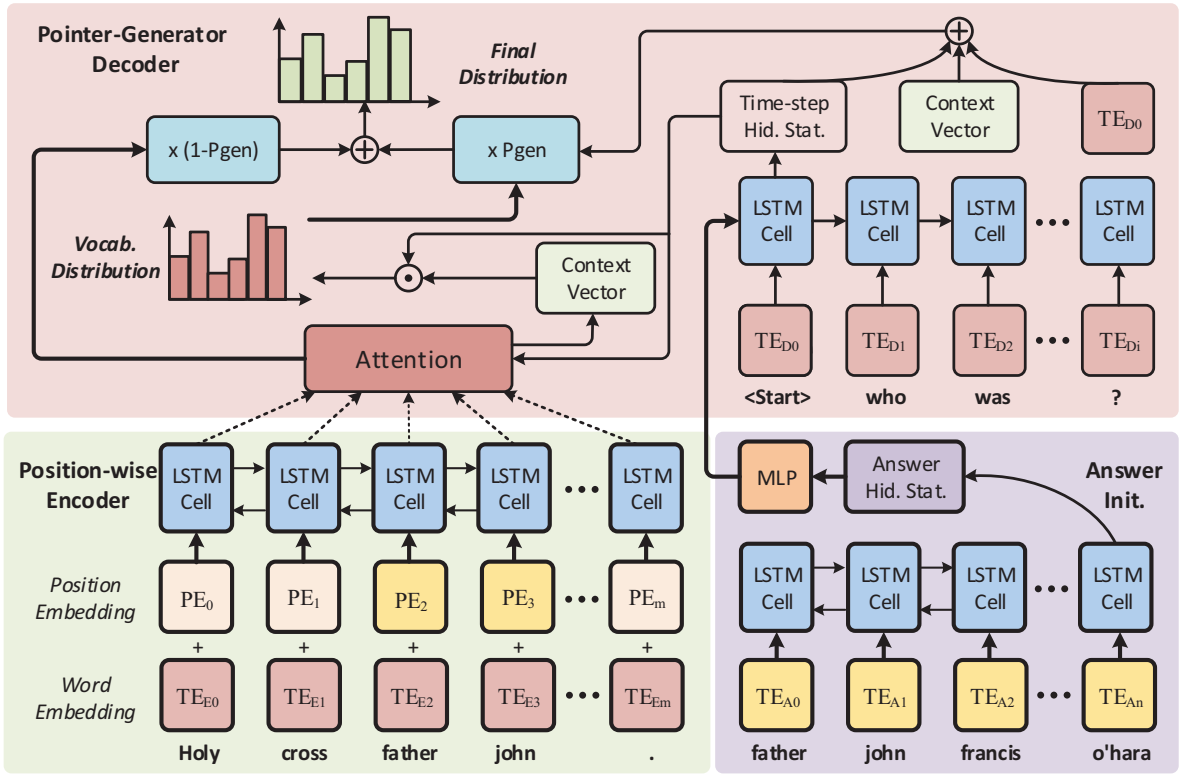

Figure 1. The APGN architecture. We use the separate answer encoder to get the representation of the answer. Then, we use it to initialize the decoder.

To solve the problem of rule-based methods, Du et al. [6] introduced a deep Seq2seq model to generate questions for the sentences extracted from reading comprehension datasets. The method achieved better performance than rule-based methods in both automatic and human evaluations. Zhou et al. [7] added lexical features (POS, NER) and answer positions to enrich the encoder. Moreover, they used copy mechanism to generate words in the source sentence. Dong et al. [8] predicted the types of the target answers by using classification model and then added to generate questions.

The end-to-end neural network can make the model self-adaptive learning and avoid complex rules. In addition, the release of large-scale reading comprehension datasets provides more corpus resources for QG task and promotes the development of the neural network-based QG methods. Our method is based on PGN which is an attention-based Seq2seq model with copy mechanism.

III. Task Definition

Given a sentence $X = \{x_1, x_2, ..., x_m\}$ and a target answer $A = \{a_1, a_2, ..., a_n\}$, A are continuous tokens of X, i.e. $A \subset X$. The QG task aims to generate a natural question Y related to the information of X and A. The QG task is defined as finding $\overline{y}$:

$$\overline{y} = \arg\max_{y} P(y|X, A) \tag{1}$$

where $P(y|X, A)$ denotes the log-likelihood conditioned on X and A.

IV. Model

We use PGN as our baseline model. In order to generate different questions to diverse target answers in one source sentence, our model incorporates target answers and combines with source sentences in the decoder. Concretely, we encode a target answer separately to get the representation. Then, we use the representation to initialize the decoder. So, target answers and source sentences work together in the decoder. Fig. 1 exhibits the model overview.

A. Baseline:Pointer-Generator Network (PGN)

1) Encoder: We use a singer-layer Bidirectional Long Short-Term Memory (Bi-LSTM) [10] to encode the source sentence X. It produces a sequence of hidden states $o = \{o_1, o_2, ..., o_m\}$ through the layer. Each hidden state concatenates the forward and the backward LSTM representation:

$$\overrightarrow{o_i} = \overrightarrow{LSTM1}(x_i^e, \overrightarrow{o_{i-1}}) \tag{2}$$

$$\overleftarrow{o_i} = \overleftarrow{LSTM1}(x_i^e, \overleftarrow{o_{i+1}}) \tag{3}$$

$$o_i = [\overrightarrow{o_i}, \overleftarrow{o_i}] \tag{4}$$

where x_i^e is word-level embedding at step i.

2) Decoder: The decoder serves as an attention-based LSTM layer with copy mechanism. At each decoding step t, the last hidden state h_{t-1} and generated word embedding x_t^e are fed to obtain current step hidden state h_t. Then we use attention mechanism [11] to compute the context vector c_t, which denotes a fixed-size representation of the importance score e_i^t between current hidden state h_t and each encoder hidden state o_i. The e_i^t are normalized by softmax function, and then get the c_t via weight sum:

$$e_i^t = v^T tanh(W_o o_i + W_h h_t + b_{attn}) \tag{5}$$

$$a^t = softmax(e^t) \tag{6}$$

$$c_t = \sum_i^m a_i^t o_i \tag{7}$$

where v^T, W_o, W_h, and b_{attn} are learnable parameters.

Then we combine the current state h_t and context vector c_t to predict next word with a softmax layer over the vocabulary:

$$P_{vocab}(W) = softmax(V^{'}(V[h_t, c_t] + b) + b^{'}) \tag{8}$$

where $V^{'}$, V, b, and $b^{'}$ are learnable parameters.

We employ copy mechanism [9] into our model. For each decoder step, p_{gen} denotes the probability of the word from the vocabulary, versus copying word from source sentence. At current step t, we use a non-linear layer with the current state h_t, context vector c_t and word embedding x_t^e to calculate p_{gen}:

$$p_{gen} = \sigma(W_c c_t + W_h^{'} h_t + W_x x_t^e + b_p) \tag{9}$$

where W_c, $W_h^{'}$, W_x, and b_p are learnable parameters, σ is sigmoid activation function.

Finally, we compute the final probability distribution with p_{gen}:

$$P(W) = p_{gen} P_{vocab}(W) + (1 - p_{gen}) \sum_{i:w_i=w} a_i^t \tag{10}$$

When the generated word is "STOP" or attain the maximum decoding length, the question is generated completely.

B. Separate Target Answer Encoder

In the model, we encode target answer A separately and use the final step hidden state as the representation of the target answer A. Then we initialize the decoder hidden state with the representation to improve the accuracy of the generated question. We use pre-trained word embedding to convert A into vectors $A^e = \{a_1^e, a_2^e, ..., a_n^e\}$ with $a_i^e \in \mathbb{R}^{d_e}$, where d_e is the dimensionality of word-level target answer embedding. To capture more context information, we use Bi-LSTM to encode A^e. Each hidden state is a concatenation of a forward and a backward LSTM representation:

$$\overrightarrow{s_i} = \overrightarrow{LSTM2}(a_i^e, \overrightarrow{s_{i-1}}) \tag{11}$$

$$\overleftarrow{s_i} = \overleftarrow{LSTM2}(a_i^e, \overleftarrow{s_{i+1}}) \tag{12}$$

Table I
THE NUMBER OF DATASETS

Dataset	Train	Dev	Test
split-1	70484	10570	11877
split-2	86607	8964	8963
MS MARCO	93267	10363	8770

$$s_i = [\overrightarrow{s_i}, \overleftarrow{s_i}] \tag{13}$$

where the Bi-LSTM does not share parameters with (2) and (3).

The final step hidden state s_n contains previous context information, so it is used to represent the target answer. In order to make full use of the target answer information, we use a fully connected layer to get deeper semantic representation:

$$new\_s_n = Relu(W_s s_n + b_s) \tag{14}$$

where W_s and b_s are learnable parameters, $Relu$ is an activation function. new_s_n denotes the new target answer representation and is used to initialize decoder hidden state.

C. Combination

Decoder Initialization We use the target answer representation new_s_n to initialize the decoder hidden state:

$$h_0 = new\_s_n \tag{15}$$

Then, the process of the decoder is modified. Because of the initialization, the baseline model PGN (5)-(10) incorporate information of target answers. Besides, the information of target answers and source sentences will work together better.

V. EXPERIMENTAL SETUP

A. Datasets

We conduct the experiments on SQuAD [12] and MS MARCO [13]. To be fair, we use the same datasets that were used by previous work. SQuAD has two division that are denoted as *split-1* and *split-2*.

SQuAD The original SQuAD consists of 536 articles from Wikipedia and more than 100k question-answer pairs posed about the articles by crowd-workers. Answers are sub-spans in the articles.

split-1 Du et al. [6] extracted sentences and paired them with the questions, and then re-divided them into train/dev/test splits. Because we use target answers, we extract them from sentences.

split-2 Zhou et al. [7] extracted sentence-answer-question triples to build train, dev and test sets.

MS MARCO In this dataset, there are 1,010,916 questions which sampled from real anonymized user queries. Each question has corresponding answers and passages from real web documents. We extract a subset of MARCO data where answer is a sub-span within the article.We use dev set as test set, and split train set into train and dev set with ratio 9:1.

We tokenize datasets with Stanford CoreNLP [14] and then lower-case them. Table I shows the number of datasets.

B. Implementation Details

We set the cutoff length of the input sentence as 100 words and the question as 30 words. We use 50,000 most frequent words that appeared in train set as vocabulary in both source and target. We use 300-dimensional pre-trained Glove [15] embedding and 16-dimensional randomly initialized position embedding. The hidden size of both the encoder and decoder is 256. During training, we use Adam [16] optimizer with learning rate 0.0005. When testing, we conduct beam search with beam size of 4.

C. Evaluation

We use the evaluation package released by Chen et al. [17]. The package includes BLEU-4 [18], METEOR [19] and ROUGE$_L$ [20].

VI. Results and Analysis

A. Main Results

To prove the effectiveness of our model, we compare it with several competitive systems. Next, we briefly introduce their approaches and experimental settings.

Du [6] is an attention-based Seq2seq model and does not use the target answer information.

NQG++ [7] is a Seq2seq model with copy mechanism. It uses Bi-GRU to encode the concatenation of word embedding, answer position and lexical features (POS, NER). Answer positions use BIO scheme.

M2S+cp [21] consists of a multi-perspective encoder and a decoder with the copy mechanism. It uses multi-perspective context matching algorithm to detect whether each source word belongs to the relevant context of the answer.

Sun [22] is a hybrid model based on Pointer-Generator network. Answer-focused model generates a question word in a restricted vocabulary of question words. Position-aware model incorporates word position embeddings to gain position-aware attention for further generation.

s2s-a-ct-mp-gsa [23] is a Seq2seq model with answer tagging, gated self-attention and maxout pointer mechanism.

Ass2s [24] is an answer-separated Seq2seq which masks answer tokens with "a" and a keyword-net which extracts key information from target answer.

Table II shows results on split-1 and split-2, and Table III is the MS MARCO result. Note that the results is the final which uses position feature. Our model performance achieves the start of the art on split-1. In addition, BLEU-4 is 0.98% higher than the PGN on MS MARCO as shown in Table III. Expect Du, other systems use position feature. In our model, we propose a simpler way to use target answer position information than other competitive systems and achieve higher performance. Note that we do not use lexical features in our model when we experiment on split-2.

Table II
PERFORMANCE COMPARISON(%).

Model	split-1			split-2
	BLEU-4	METEOR	ROUGE$_L$	BLEU-4
Du	12.28	16.62	39.75	-
NQG++	-	-	-	13.29
M2S+cp	13.98	18.77	42.72	13.91
Sun	-	-	-	15.64
s2s-a-ct-mp-gsa	15.32	19.29	43.91	15.82
Ass2s	16.20	19.92	43.96	**16.17**
APGN	**17.05**	**20.53**	**44.06**	15.16

Table III
THE PERFORMANCE ON MS MARCO(%)

	BLEU-4	METEOR	ROUGE$_L$
PGN	8.18	22.15	35.68
APGN	**9.16**	**23.61**	**38.21**

B. Discussion

1) Impact of A Source Sentence with Different Target Answers: In this section, we statistics the number of different target answers in the one source sentence as shown in Table IV. "Difference" represents the number of target answers in one source sentence. "Count" represents the number of source sentences. In the test dataset, there are almost 70% source sentences that have different target answers.

Table IV
THE NUMBER OF DIFFERENT TARGET ANSWERS IN ONE SENTENCE

Difference	Count	Difference	Count
1	3557	7	21
2	1882	8	12
3	683	9	4
4	304	10	2
5	143	12	1
6	42	13	1

To verify the effectiveness of our method, we calculate the BLEU-4 scores of PGN and our model towards the source sentences mentioned above. Our model gains 34.54% BLEU-4 score which is 7.06% higher than PGN

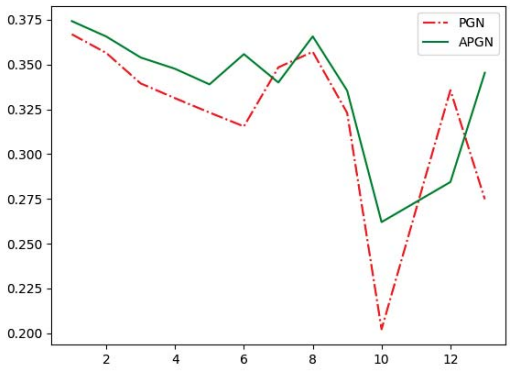

Figure 2. The BLEU-4 scores

(27.48%). To make it clearer, we draw a line chart as shown in Fig. 2. Generally, our model is higher than the PGN. It illustrates our proposed method can generate different questions to a source sentence with diverse target answers.

2) Reasonability of Separate Target Answer Encoder to Initialize the Decoder: To prove the reasonability of decoder initialization with a separate target answer encoder, we design the following experiments. Firstly, we encode target answers sharing parameters with source sentences encoder. Secondly, we add the representation of the target answer in the decoding step except for the first. The results show in Table V.

Table V
THE BLEU-4 SCORES (%)

Model	BLEU-4
PGN	15.23
PGN+share	15.60
PGN+add_init	1.07
APGN	**16.13**

As shown in Table V, the separate encoder is better than the sharing parameters one. The reason is that encoding separately makes the model more focused on the target answer, thus generating its corresponding question. For initialization, only initialize the first state of the decoder is valid. We think adding the representation of the target answer at each step can greatly mislead the generation process, resulting in lower performance.

3) Question Words: In this section, we analyse the distribution of generated question words. A question word represents the type of a target answer in one source sentence. We select common question words from split-1 test set and statistic proportion in target questions (TGT), PGN and APGN. As shown in Table VI, it can be observed that our APGN is closer to the percentage of question words on TGT. To make it clearer, we draw a chart (see Fig. 3) that shows the difference of question words number between APGN, PGN and TGT respectively.

Table VI
THE PERCENTAGE OF QUESTION WORDS ON TGT, PGN AND APGN

Question Word	TGT(%)	PGN(%)	APGN(%)
What	56.41	65.03	62.81
How	12.15	11.18	12.55
Who	11.31	10.31	14.06
Which	8.55	0.15	1.27
When	5.57	7.54	4.61
Where	3.76	5.54	3.92
Why	0.78	0.18	0.24
Others	1.47	0.07	0.54

As shown in Fig. 3, expect for "who" type, the difference of question words number between APGN and TGT is smaller than that between PGN and TGT. It further illustrates incorporating the target answer helps to capture the question word corresponding to the target answer, thus ensuring the generated question is used to ask the target answer.

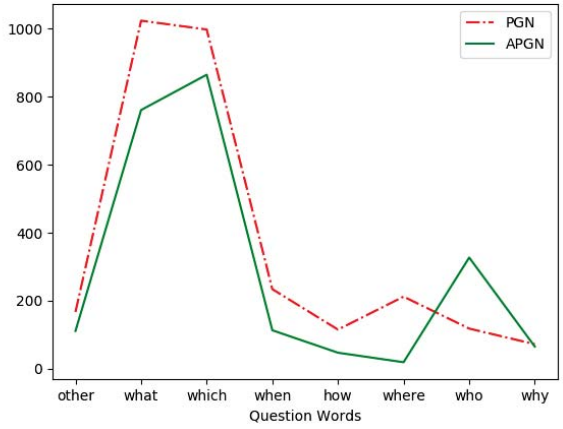

Figure 3. The difference of question words number between PGN, APGN and TGT respectively.

4) Impact of Generation Process: In general, the first token of a question is a question word. In the last section, we have proved our proposed method improves the accuracy of question words. In this section, we analyse the impact of question words to subsequence words of the generation process.

We force the first generated word to be correct in the decoder for baseline model PGN, then we evaluate those generated questions. We find the BLEU-4 is 2.27% higher than PGN. Thus, once question words are generated correctly, the following process of generation will be better.

To some extent, our proposed method has a good effect on the subsequence generation process by improving the accuracy of question words. We think this is because we have determined the type of question by the initialization of the target answer. Because of the characteristic of the LSTM, correctness will be propagated in the following generation.

5) Position Feature: In the model, we concatenate word-level embedding with position embedding. We use "01" shame to label words in the source sentences. Label *1* denotes the word is in the target answer. In contrast, label *0* denotes the word is not in the target answer. Then, the answer position labels are embedded in real-valued vectors with the dimensionality of 16.

VII. CONCLUSION

In this paper, we propose encoding separately target answers to get the representation. Then, we use this representation of the target answer to initialize the decoder. Through analysis of generated questions, we find our proposed method can generate different questions to a source sentence that has different target answers. Besides, we describe the reasonability of a separate target answer encoder to initialize the decoder. We also find our method can generate question words more correctly than PGN. Once the question word is generated correctly, the subsequence of the question will be better.

ACKNOWLEDGMENT

This research work is supported by National Natural Science Foundation of China (Grants No.61672368, No.61703293 and 2017YFB1002104). The authors would like to thank the anonymous reviewers for their insightful comments and suggestions. Yu Hong, Professor Associate in Soochow University, is the corresponding author of the paper, whose email address is tianxianer@gmail.com.

REFERENCES

[1] V. Rus, B. Wyse, P. Piwek, M. Lintean, S. Stoyanchev, and C. Moldovan, "The first question generation shared task evaluation challenge," in *Proceedings of the 6th International Natural Language Gen-eration Conference.* Association for Computational Linguistics, 2010, pp. 251–257.

[2] M. Heilman and N. A. Smith, "Good question! statistical ranking for question generation," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 2010, pp. 609–617.

[3] J. Weizenbaum, "Eliza—a computer program for the study of natural language communication between man and machine," *Communications of the ACM*, vol. 9, no. 1, pp. 36–45, 1966.

[4] D. Lindberg, F. Popowich, J. Nesbit, and P. Winne, "Generating natural language questions to support learning online," in *Proceedings of the 14th European Workshop on Natural Language Generation*, 2013, pp. 105–114.

[5] M. Liu, V. Rus, and L. Liu, "Automatic chinese factual question generation," *IEEE Transactions on Learning Technologies*, vol. 10, no. 2, pp. 194–204, 2017.

[6] X. Du, J. Shao, and C. Cardie, "Learning to ask: Neural question generation for reading comprehension," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2017, pp. 1342–1352.

[7] 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.

[8] X. Dong, Y. Hong, X. Chen, W. Li, M. Zhang, and Q. Zhu, "Neural question generation with semantics of question type," in *CCF International Conference on Natural Language Processing and Chinese Computing*. Springer, 2018, pp. 213–223.

[9] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2017, pp. 1073–1083.

[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

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

[12] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ questions for machine comprehension of text," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2016, pp. 2383–2392.

[13] T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng, "Ms marco: A human generated machine reading comprehension dataset," in *Proceedings of 30th Conference on Neural Information Processing Systems*, 2016.

[14] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *Association for Computational Linguistics (ACL) System Demonstrations*, 2014, pp. 55–60.

[15] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[16] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[17] X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollár, and C. L. Zitnick, "Microsoft coco captions: Data collection and evaluation server," *arXiv preprint arXiv:1504.00325*, 2015.

[18] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2002, pp. 311–318.

[19] M. Denkowski and A. Lavie, "Meteor universal: Language specific translation evaluation for any target language," in *Proceedings of the ninth workshop on statistical machine translation*, 2014, pp. 376–380.

[20] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," *Text Summarization Branches Out*, 2004.

[21] L. Song, Z. Wang, W. Hamza, Y. Zhang, and D. Gildea, "Leveraging context information for natural question generation," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, 2018, pp. 569–574.

[22] 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.

[23] 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.

[24] Y. Kim, H. Lee, J. Shin, and K. Jung, "Improving neural question generation using answer separation," *arXiv preprint arXiv:1809.02393*, 2018.