A Recurrent BERT-based Model for Question Generation

Ying-Hong Chan
Department of Computer Science
National Chung Hsing University
Taichung, Taiwan
harry831120@gmail.com

Yao-Chung Fan
Department of Computer Science
National Chung Hsing University
Taichung, Taiwan
yfan@nchu.edu.tw

Abstract

In this study, we investigate the employment of the pre-trained BERT language model to tackle question generation tasks. We introduce three neural architectures built on top of BERT for question generation tasks. The first one is a straightforward BERT employment, which reveals the defects of directly using BERT for text generation. Accordingly, we propose another two models by restructuring our BERT employment into a sequential manner for taking information from previous decoded results. Our models are trained and evaluated on the recent question-answering dataset SQuAD. Experiment results show that our best model yields state-of-the-art performance which advances the BLEU 4 score of the existing best models from 16.85 to 22.17.

1 Introduction

Question generation (QG) problem, which takes a context text and an answer phase as input and generates a question corresponding to the given answer phase, has received tremendous interests in recent years from both industrial and academic natural language processing communities (Zhao et al., 2018; Zhou et al., 2017; Du et al., 2017). The state-of-the-art model mainly adopts neural QG approaches: training a neural network based on sequence-to-sequence framework. So far, the best performing result is reported in (Zhao et al., 2018), which advances the state-of-the-art results from 13.9 to 16.85 (BLEU 4).

The existing QG models mainly rely on recurrent neural networks (RNN), e.g. long short-term memory LSTM network (Hochreiter and Schmidhuber, 1997) or gated recurrent unit (Chung et al., 2014), augmented by attention mechanisms (Luong et al., 2015). However, the inherent sequential nature of the RNN models suffers from the problem of handling long sequences. Therefore, the existing QG models (Du et al., 2017; Zhou et al., 2017) mainly use only sentence-level information as a context text for question generation. When applied to a paragraph-level context, the existing models show significant performance degradation. However, as indicated by (Du et al., 2017), providing paragraph-level information can improve QG performance. For handling long context, the work (Zhao et al., 2018) introduces a maxout pointer mechanism with a gated self-attention encoder for processing paragraph-level input. The work reports state-of-the-art performance.

Recently, the NLP community has seen the excitement around neural learning models that make use of pre-trained language models (Devlin et al., 2018; Radford et al., 2018). The latest development is BERT, which has shown significant performance improvement over various natural language understanding tasks, such as document summarization, document classification, etc.

Given the success of the BERT model, a natural question follows: can we leverage the BERT models to further advance the state-of-the-art for QG tasks? By our study, the answer is yes. Intuitively, the BERT employment brings two advantages for tackling the QG problem. First, as reported by studies (Devlin et al., 2018; Radford et al., 2018), employing pre-training language models has shown to be effective for improving NLP tasks. Second, the BERT model is a stack of multi-layer Transformer block (Vaswani et al., 2017), which eschews recurrence structure and relies entirely on self-attention mechanism to draw global dependencies between input sequences. With the Transformer blocks, processing paragraph-level contexts for QG are therefore to be possible.

In this study, we investigate the employment of the pre-trained BERT language model to tackle question generation tasks. We introduce three neural architectures built on top of BERT for question generation tasks. The first one is a straightforward

BERT employment, which reveals the defects of directly using BERT for text generation. As will be shown in the experiment, the naive BERT employment (called BERT-QG, BERT Question Generation) offers poor performance, as by construction, BERT produces all tokens at a time without considering decoding results in previous steps. We find that the question generated by the naive employment is not even a readable sentence. As a result, we propose a sequential question generation model based on BERT as our second model called BERT-SQG (BERT-Sequential Question Generation) for taking information from previous decoded results. As will shown in the performance evaluation, the BERT-SQG model outperforms the exiting best model (Zhao et al., 2018) by advancing the state-of-the-art results from 16.85 to 21.04 (BLEU 4).

Furthermore, we propose an augmented model called BERT-HLSQG (Highlight Sequential Question Generation) for further enhancing the performance of the BERT-SQG. Our BERT-HLSQG model works by marking the answer with `[HL]` tokens to avoid possible ambiguity in specifying answers for question generation. Such design further improves the BLEU 4 score from 21.04 to 22.17.

The contribution of this paper is summarized as follows.

- In this paper, we investigate the employment of using the BERT model for QG tasks. We show that the sequential structure is important for the decoding of text generation. Aiming at this point, we propose two sequential question generation models based on BERT in this paper.

- Furthermore, we propose a simple but effective input encoding scheme, which inserts special highlighting tokens `[HL]` before and after the given answer span, to address the ambiguity issue when an answer phase appears multiple times in the question.

- Extensive experiments are conducted using benchmark datasets, and the experiment results show the effectiveness of our question generation model. Our model outperforms the existing best models (Zhao et al., 2018) and pushes the state-of-the-art result from 16.85 to 22.17 (BLEU 4).

The rest of this paper is organized as follows. In Section 2, we discuss the related work for QG generation. In Section 3, we review the BERT model (the basic building block for our model). In Section 4, we introduce our models for question generation, and Section 5 provides the experiment results. In Section 6, we conclude the paper and discuss future work.

2 Related Work

The question generation has been mainly tackled with two types of approaches. One is built on top of heuristic rules that creates questions with manually constructed template and ranks the generated results (Heilman and Smith, 2010; Mazidi and Nielsen, 2014; Labutov et al., 2015). In (Labutov et al., 2015), the authors propose to use a crowdsourcing policy to generate question templates from a large amount of text to generate question. The research in (Heilman and Smith, 2010) proposes to use manually written rules to perform a sequence of general-purpose syntactic transformations to turn declarative sentences into questions. The generated questions are then ranked by a logistic regression model to select the qualified questions for later use. And, the research in (Yao et al., 2012) proposes to convert the sentence into a Minimal Recursion Semantics (MRS) representation through linguistic parsing, and then construct semantic structures and grammar rules from the representation to generate questions through the manually designed rules. Those approaches heavily depend on human effort, which makes them hard to scale up and being generalized in various domains.

The other one, which is becoming increasingly popular, is to train an end-to-end neural network from scratch by using sequence to sequence or encoder-decoder framework, e.g. (Du et al., 2017; Yuan et al., 2017; Song et al., 2017; Zhou et al., 2017; Zhao et al., 2018).

(Du et al., 2017) pioneered the work of automatic QG tasks using an end-to-end trainable seq2seq neural model. Automatic and human evaluation results showed that the proposed model outperformed the previous rule-based systems (Heilman and Smith, 2010; Rus et al., 2010). However, in their study, there was no control about which part of the context text the generated question was asking about.

On the other hands, the work (Zhou et al., 2017;

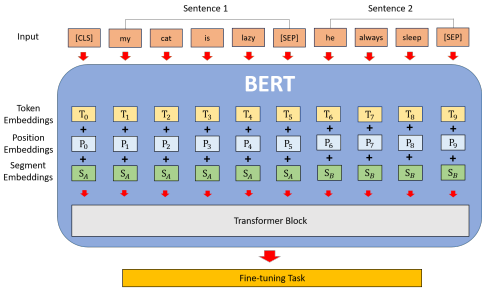

Figure 1: BERT input architecture. Input Transformer block embedding is the sum of the three embeddings, and then use the hidden vector to fine tune each task.

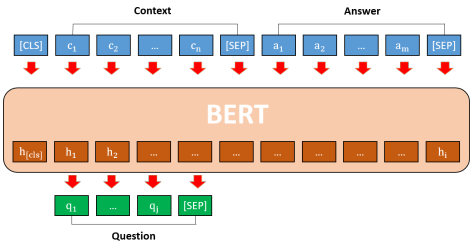

Figure 2: The BERT-QG architecture

Subramanian et al., 2017; Yuan et al., 2017) propose to encode answer location information using an annotation vector corresponding to the answer word positions. (Zhou et al., 2017) utilized rich features of the passage including answer positions. (Subramanian et al., 2017) deployed a two-stage neural model that detects important phrases and accordingly generates questions conditioned on the important phases. (Yuan et al., 2017) combined supervised and reinforcement learning in the training of their model using policy gradient techniques to maximize several rewards that measure question quality. Instead of using an annotation vector to tag the answer locations, the (Song et al., 2017) propose to employ a unified framework for QG and question answering by encoding both the answer and the passage with a multi-perspective matching mechanism. Further, (Tang et al., 2017; Wang et al., 2017) proposed joint models to address QG and question answering as a multi-task learning setting. (Duan et al., 2017) conducted QG for improving question answering. Due to the mixed objectives including question answering, the performance reported by their work was lower than the state-of-the-art results. In (Zhao et al., 2018), authors propose a maxout pointer mechanism with a gated self-attention encoder to solve the problem of processing long context for question generation.

All above-mentioned models are RNN base models, which suffers from the issue of processing long context/sequences. Compared with the RNN based model, our models based on BERT composed by transformer models (Vaswani et al., 2017). As shown in the later section, the question generated by our model is more semantically coherent and fluent.

3 BERT Overview

The BERT model is built by a stack of multi-layer bidirectional Transformer encoder (Vaswani et al., 2017). The BERT model has three architecture parameter settings: the number of layers (i.e., transformer blocks), the hidden size, and the number of self-attention heads in a transformer block.

For using BERT model, the input is required to be aligned as the BERT's specific input sequence. In general, a special token `[CLS]` is inserted as the first token for BERT's input sequence. The final hidden state of the `[CLS]` token is designed to be used as a final sequence representation for classification tasks. The input token sequence can be a pack of multiple sentences. To distinguish the information from different sentences, a special token `[SEP]` is added between the tokens of two consecutive sentences. In addition, a learned embedding is added to every token to denote whether it belongs to which sentence. For example, given a sentence pair (s_i, s_j) where s_i contains $|s_i|$ tokens and s_j contains $|s_j|$ tokens, the BERT input sequence is formulated as a sequence in the following form:

$$X = (\texttt{[CLS]}, t_{i,1}, ..., t_{i,|s_i|}, \texttt{[SEP]}, t_{j,1}..., t_{j,|s_j|})$$

As shown in Figure 1, the input representation of a given token is the sum of three embeddings: the token embeddings, the segmentation embeddings, and the position embeddings. Then the input representation is fed forward into extra layers to perform a fine-tuning procedure. The BERT model can be employed in three language modeling tasks: sequence-level classification, span-level prediction, and token-level prediction tasks. The fine-tuning procedure is performed in a task-specific manner. The details of our fine-tuning procedure are introduced in the later subsections.

4 BERT for Question Generation

In the following subsections, we introduce our models for QG. In Subsection 4.1, we introduce the naive BERT employment (BERT-QG), which serves as a first cut for using BERT for QG. BERT-QG offer poor performance but draws some insights for using BERT in QG tasks. Further, in Subsection 4.2, we introduce BERT-SQG by considering sequential information when generating questions. Last, in Subsection 4.3, we introduce BERT-HLSQG which shows the SOTA results for QG based on BERT.

4.1 BERT-QG

As an initial attempt, we first adapt the BERT model for QG as follows. First, for a given context paragraph $C = [c_1, ..., c_{|C|}]$ and an answer phase $A = [a_1, ..., a_{|A|}]$, the input sequence X is aligned as

$$X = (\texttt{[CLS]}, C, \texttt{[SEP]}, A, \texttt{[SEP]})$$

Let $\mathrm{BERT}()$ be the BERT model. We first obtain the hidden representation $\mathbf{H} \in \mathbb{R}^{|X| \times h}$ by $\mathbf{H} = \mathrm{BERT}(X)$, where $|X|$ is the length of the input sequence and h is the size of the hidden dimension. Then, $\mathbf{H}$ is passed to a dense layer $\mathbf{W} \in \mathbb{R}^{h \times |V|}$ followed by a softmax function as follows.

$$Pr(w|x_i) = softmax(\mathbf{H} \cdot \mathbf{W} + \mathbf{b}), \forall x_i \in X$$

$$\hat{q}_i = \mathrm{argmax}_w Pr(w|x_i)$$

The softmax is applied along the dimension of the sequence. All of the parameters of BERT and $\mathbf{W}$ are fine-tuned jointly to maximize the log-probability of the correct token q_i. The model architecture is shown in Figure 2. As such, a sequence of tokens $[w_1, ..., w_{|x|}]$ is generated and we use the first generated `[SEP]` symbol as the end of the generated question sentence.

4.2 BERT-SQG

In text generation tasks, as proposed by (Sutskever et al., 2014), considering the previous decoded results has significant impacts on the quality of the generated text. However, in BERT-QG, the token generation is performed without previous decoded result information. Due to this consideration, we propose a sequential question generation model based on BERT (called BERT-SQG).

In BERT-SQG, we take into consideration the previous decoded results for decoding a token. We adapt the BERT model for question generation as follows. First, for a given context paragraph $C = [c_1, ..., c_{|C|}]$ and an answer phase $A = [a_1, ..., a_{|A|}]$, and $\hat{Q} = [\hat{q}_1, ..., \hat{q}_i]$ the input sequence X_i is formulated as

$$X_i = (\texttt{[CLS]}, C, \texttt{[SEP]}, A, \texttt{[SEP]}, \hat{q}_1, ..., \hat{q}_i, \texttt{[MASK]})$$

Then, the input sequence X_i is represented by the BERT embedding layers and then travel forward into the BERT model. After that, we take the final hidden state (i.e., the output of the Transformer blocks) for the last token `[MASK]` in the input sequence. We denote the final hidden vector of `[MASK]` as $\mathbf{h}_{\texttt{[MASK]}} \in \mathbb{R}^h$. We adapt BERT model by adding an affine layer $\mathbf{W}_{\texttt{SQG}} \in \mathbb{R}^{h \times |V|}$ to the output of the `[MASK]` token. We compute the label probabilities $Pr(w|X_i) \in R^{|V|}$ by a softmax function as follows.

$$Pr(w|X_i) = softmax(\mathbf{h}_{\texttt{[MASK]}} \cdot \mathbf{W}_{\texttt{SQG}} + \mathbf{b}_{\texttt{SQG}})$$

$$\hat{q}_i = \mathrm{argmax}_w Pr(w|X_i)$$

Subsequently, the newly generated token $\hat{q}_i$ is appended into X and the question generation process is repeated (as illustrated in Figure 3) with the new X until `[SEP]` is predicted. We report the generated tokens as the predicted question. In Table 1, we give an example of the actual running of the model.

4.3 BERT-HLSQG

In BERT-SQG, we find there are two shortcomings for producing quality results. First, when processing lengthy context, we find that the generated question is often with lower quality. Second, when an answer phase appears multiple times in the context, there is ambiguity for select which one to generate questions. As a result, poor results are reported when we use the BLEU score for performance evaluation. To address these shortcomings, we propose to further restructure BERT-SQG as follows. First, for a given context paragraph $C = [c_1, ..., c_{|C|}]$ and an answer phase $A = [a_1, ..., a_{|A|}]$, we integrate C and A into a new C' in the following form.

$$C' = [c_1, c_2, ..., \texttt{[HL]}, a_1, ..., a_{|A|}, \texttt{[HL]}, ..., c_{|C|}]$$

	X	x_i
iter0	[CLS] The Super Bowl 50 was played at Santa Clara, California. [SEP] Santa Clara, California. [SEP] [MASK]	Where
iter1	[CLS] The Super Bowl 50 was played at Santa Clara, California. [SEP] Santa Clara, California. [SEP] Where [MASK]	did
iter2	[CLS] The Super Bowl 50 was played at Santa Clara, California. [SEP] Santa Clara, California. [SEP] Where did [MASK]	Super
iter3	[CLS] The Super Bowl 50 was played at Santa Clara, California. [SEP] Santa Clara, California. [SEP] Where did Super [MASK]	Bowl
iter4	[CLS] The Super Bowl 50 was played at Santa Clara, California. [SEP] Santa Clara, California. [SEP] Where did Super Bowl [MASK]	50
iter5	[CLS] The Super Bowl 50 was played at Santa Clara, California. [SEP] Santa Clara, California. [SEP] Where did Super Bowl 50 [MASK]	take
iter6	[CLS] The Super Bowl 50 was played at Santa Clara, California. [SEP] Santa Clara, California. [SEP] Where did Super Bowl 50 take [MASK]	place?
iter7	[CLS] The Super Bowl 50 was played at Santa Clara, California. [SEP] Santa Clara, California. [SEP] Where did Super Bowl 50 take place [MASK]	[SEP]
iter8	[CLS] The Super Bowl 50 was played at Santa Clara, California. [SEP] Santa Clara, California. [SEP] Where did Super Bowl 50 take place [SEP] [MASK]	

Table 1: BERT-SQG Running Example

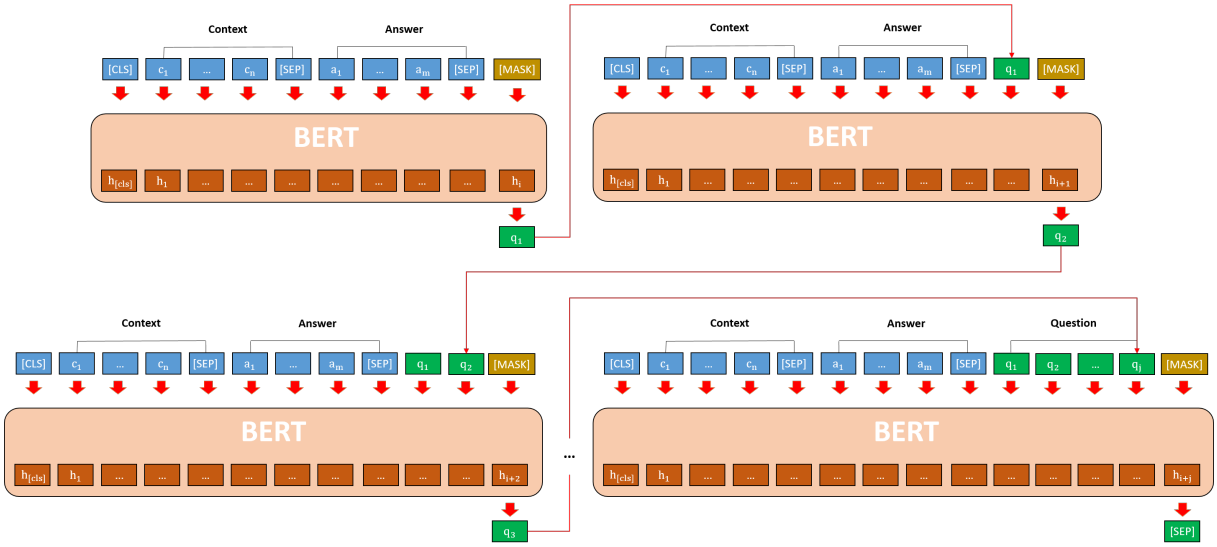

Figure 3: The BERT-SQG architecture

In C', we design and insert a new token (i.e., `[HL]`) to indicate the answer phase in the context. The observation for doing so is that we observe that for a long context, the answer phase often appears multiple times in the context, which causes ambiguity for the model for knowing which one as a target to generate question sentence. Thus, we design `[HL]` token to avoid possible ambiguity. With C', the input sequence X can be formulated as

$$X_i = (\texttt{[CLS]}, C', \texttt{[SEP]}, \hat{q}_1, ..., \hat{q}_i, \texttt{[MASK]})$$

Figure 4 shows the BERT-HLSQG model architecture. At each iteration, for generating q_i, we take the final hidden state vector $\mathbf{h}_{\texttt{[MASK]}} \in \mathbb{R}^h$ of the last token `[MASK]` in the input sequence. and connect it to an affine layer $\mathbf{W}_{\texttt{HLSQG}} \in \mathbb{R}^{h \times |V|}$. We compute the label probabilities $Pr(w|X_i) \in R^{|V|}$ by a softmax function as follows.

$$Pr(w|X_i) = softmax(\mathbf{h}_{\texttt{[MASK]}} \cdot \mathbf{W}_{\texttt{HLSQG}} + \mathbf{b}_{\texttt{HLSQG}})$$

$$\hat{q}_i = \text{argmax}_w Pr(w|X_i)$$

We show a running example of BERT-HLSQG in Table 2.

5 Performance Evaluation

In this section, we present the performance evaluation results on the QG task on SQuAD (Rajpurkar et al., 2016) dataset.

5.1 Datasets

The SQuAD dataset contains 536 Wikipedia articles and 100K reading comprehension questions (and the corresponding answers) posed about the articles. Answers of the questions are text spans in the articles.

We use the same data split settings as the previous work on the QG tasks (Du et al., 2017; Zhao et al., 2018) to directly compare the state-of-the-art results on QG tasks. Table 3 summarizes statistics for the compared datasets.

- **SQuAD 73K** In this set, we follow the same setting as (Du et al., 2017); the accessible parts of the SQuAD training data are randomly divided into a training set (80%), a development set (10%), and a test set (10%). We report results on the 10% test set.

	X	x_i
iter0	[CLS] The Super Bowl 50 was played at [HL] Santa Clara, California [HL] . [SEP][MASK]	Where
iter1	[CLS] The Super Bowl 50 was played at [HL] Santa Clara, California [HL] . [SEP] Where [MASK]	did
iter2	[CLS] The Super Bowl 50 was played at [HL] Santa Clara, California [HL] . [SEP] Where did [MASK]	Super
iter3	[CLS] The Super Bowl 50 was played at [HL] Santa Clara, California [HL] . [SEP] Where did Super [MASK]	Bowl
iter4	[CLS] The Super Bowl 50 was played at [HL] Santa Clara, California [HL] . [SEP] Where did Super Bowl [MASK]	50
iter5	[CLS] The Super Bowl 50 was played at [HL] Santa Clara, California [HL] . [SEP] Where did Super Bowl 50 [MASK]	take
iter6	[CLS] The Super Bowl 50 was played at [HL] Santa Clara, California [HL] . [SEP] Where did Super Bowl 50 take [MASK]	place?
iter7	[CLS] The Super Bowl 50 was played at [HL] Santa Clara, California [HL] . [SEP] Where did Super Bowl 50 take place [MASK]	[SEP]
iter8	[CLS] The Super Bowl 50 was played at [HL] Santa Clara, California [HL] . [SEP] Where did Super Bowl 50 take place [SEP] [MASK]	

Table 2: BERT-HLSQG Running Example

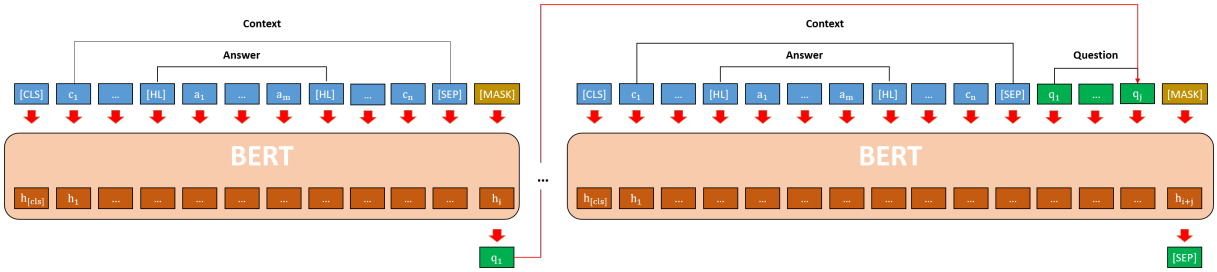

Figure 4: The BERT-HLSQG architecture

	Train	Test	Dev
SQuAD 73K	73240	11877	10570
SQuAD 81K	81577	8964	8964

Table 3: Dataset statistics: SQuAD 73K is the setting of (Du et al., 2017), and SQuAD 81K is the setting of (Zhao et al., 2018).

- **SQuAD 81K** In this set, we follow the same setting as (Zhao et al., 2018); the accessible SQuAD development data set is divided into a development set (50%), and a test set (50%). We report results on the 50% test set.

5.2 Performance Metrics

We use the evaluation package released by (Sharma et al., 2017). The package includes BLEU 1, BLEU 2, BLEU 3, BLEU 4 (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2014) and ROUGE (Lin, 2004) evaluation scripts. BLEU measures the average n-gram precision on a set of reference sentences, with a penalty for overly short sentences. BLEU-n is a BLEU score variant that uses up to n-grams for counting co-occurrences. METEOR is a recall-oriented metric, which computes the similarity between the generated sentences and ground truth sentences by considering synonyms, stemming and paraphrases. ROUGE is commonly employed to evaluate n-grams recall of the summaries with gold standard sentences as references. ROUGE-L (measured based on the longest common subsequence) results are reported.

5.3 Implementation Details

We use the PyTorch version of BERT [1] to train our BERT-QG, BERT-SQG and BERT-HLSQG models. The pre-trained model uses the officially provided $\textbf{BERT}_{\textbf{base}}$ model (12 layers, 768 hidden dimensions, and 12 attention heads.) with a vocab of 30522 words. Dropout probability is set to 0.1 between transformer layers. The Adamax optimizer is applied during the training process, with an initial learning rate of 5e-5. The batch size for the update is set at 28. All our models use two TITAN RTX GPUs for 5 epochs training. We use Dev. data for epoch model to make predictions and select the highest accuracy rate as our score evaluation model. Also, in our BERT-SQG and BERT-HLSQG model, we use the Beam Search strategy for sequence decoding. The beam size is set to 3.

5.4 Model Comparison

In this paper, we compare our models with the best performing models (Du et al., 2017; Zhao et al., 2018) in the literature. The compared models in the experiment are:

- **NQG-RC** (Du et al., 2017): A seq2seq question generation model based on bidirectional LSTMs.

- **PLQG** (Zhao et al., 2018): A seq2seq network which contains a gated self-attention encoder and a maxout pointer decoder to en-

[1] https://github.com/huggingface/pytorch-pretrained-BERT

able the capability of handling long text input. The PLQG model is the state-of-the-art models for QG tasks.

5.5 Quantitative Results

Table 5 shows the comparison results using sentence-level context texts and Table 6 shows the results on paragraph-level context. We compare the models using standard metric BLEU, ROUGE-L, and METEOR.

We have the following findings to note about the results. First, as can be observed, BERT-QG offers poor performance. The performance of BERT-QG is far from the results by other models. This result is expected as BERT-QG generates the sentences without considering the previous decoded results. However, when taking into account the previous decoded results (BERT-SQG), we effectively utilize the power of BERT and yield the state-of-the-art result compared with the existing RNN variants for QG. Also, we see that BERT-HLSQG successfully address the limitation of BERT-SQG. As shown in Table 5, BERT-HLSQG outperforms the existing best performing model by 4-5% on both benchmark datasets.

Second, the results in Table 6 further show that BERT-SQG successfully processes the paragraph-level contexts and further push the state-of-the-art from 16.85 to 21.04 in terms of BLEU 4 score. Note that NQG-RC and PLQG both use the RNN architecture, and the RNN-based models all suffer from the issue of consuming long text input. We see that the BERT model based on Transformer blocks effectively addresses the issue of processing long text. In addition, the improvement of BERT-HLSQG is more obvious under paragraph-level, which advances the score from 21.04 to 22.17 in terms of BLEU 4 score. Again, this result validates that our BERT-HLSQG model does improve the shortcomings of BERT-SQG and achieves the best score at the paragraph-level context.

5.6 Evaluation Result on Reading Comprehension Task

One issue we find in our performance evaluation is that we observe questions generated by our models are good but with a very low BLEU score. The problem for this result comes from that BLEU score is token-basis; the generated question is compared with a golden standard based on the token similarity. A question might be expressed in different ways (but semantically the same); there are many different ways of describing the same thing/question. We think the score computed based on tokens can not truly reflect the performance of our model.

In order to demonstrate the effectiveness of our model, we further evaluate our model through reading comprehension (RC) tasks. Given a context and a question, a reading comprehension task returns the answer span to the question from the given context. In this experiment, we compare and examine the impact of the question sentences generated by the BERT-SQG and BERT-HLSQG models on the RC task to further validate our model.

5.6.1 Implementation Details

In this set of experiments, our goal is to examine the difference between using human-generated questions and questions generated by our QG models to train a reading comprehension model. Specifically, we use the training data set provided by the SQuAD and divided the training data set into QG set (50%) and RC set (50%). Then, we train BERT-SQG and BERT-HLSQG models using QG sets. The model is then used to generate questions to generate the RC-SQG and RC-HLSQG sets. Finally, we use RC, RC-SQG and RC-HLSQG sets for reading comprehension task training, and compare Exact Match and F1 score with the RC model (the one trained by RC set).

Our RC model is also implemented based on the PyTorch version BERT model and fine-tuned on the officially $\text{BERT}_{\textbf{base}}$ pre-training model. The dropout rate is set to 0.1 for all Transformer layers. The optimizer is performed using AdamW, with an initial learning rate of 3e-5. The batch size for the update is set at 8. All RC models use two TITAN RTX GPUs for 2 epochs training.

5.6.2 Results and Analysis

Table 4 shows the human question and generated question experiment comparison results. We observe that the RC-SQG and RC-HLSQG data sets generated using the model for question generation differed only 4-5% from the results of the human question data set on the Exact Match and the F1 Score is only 3-4%. The average token on the question is also close to the human question. These results demonstrate that the quality of the problems generated by our model is close to hu-

	Exact Match	F1 score	Question avg. tokens
RC	79.09	86.82	12.29
RC-SQG	74.07	82.91	12.09
RC-HLSQG	74.36	83.07	12.06

Table 4: Reading comprehension evaluation results

	Model	**BLEU 1**	**BLEU 2**	**BLEU 3**	**BLEU 4**	**METEOR**	**ROUGE-L**
SQuAD 73K	NQG-RC	43.09	25.96	17.50	12.28	16.62	39.75
	PLQG	43.47	28.23	20.40	15.32	19.29	43.91
	BERT-QG	34.17	15.52	8.36	4.47	14.78	37.60
	BERT-SQG	48.38	**33.15**	24.75	19.08	22.43	46.94
	BERT-HLSQG	**48.29**	33.12	**24.78**	**19.14**	**22.89**	**47.07**
SQuAD 81K	PLQG	44.51	29.07	21.06	15.82	19.67	44.24
	BERT-QG	34.18	15.51	8.57	4.97	14.57	37.65
	BERT-SQG	50.18	35.03	26.60	20.88	23.84	48.37
	BERT-HLSQG	**50.71**	**35.44**	**26.95**	**21.20**	**24.02**	**48.68**

Table 5: Comparison between our model and the published methods using sentence level context

	Model	**BLEU 1**	**BLEU 2**	**BLEU 3**	**BLEU 4**	**METEOR**	**ROUGE-L**
SQuAD 73K	NQG-RC	42.54	25.33	16.98	11.86	16.28	39.37
	PLQG	45.07	29.58	21.60	16.38	20.25	44.48
	BERT-QG	37.49	18.32	10.47	6.10	16.80	41.01
	BERT-SQG	**50.00**	34.54	25.98	20.11	23.88	48.12
	BERT-HLSQG	49.73	**34.60**	**26.13**	**20.33**	**23.88**	**48.23**
SQuAD 81K	PLQG	45.69	30.25	22.16	16.85	20.62	44.99
	BERT-QG	32.61	14.50	7.70	4.08	14.18	37.94
	BERT-SQG	50.89	35.49	26.87	21.04	24.25	48.66
	BERT-HLSQG	**51.54**	**36.45**	**27.96**	**22.17**	**24.80**	**49.68**

Table 6: Comparison between our model and the published methods using paragraph level context

mans, and the use of reading comprehension tasks also has effective.

6 Conclusion

In this paper, we propose models that generate a question from the input context (sentence or paragraph) and the target answer based on BERT models. Our models are transformer models which can handle long-term dependencies well. To make the generation process sequential, we propose to restructure our model to generate one word at a time, using the encoded task inputs and the previously generated words as inputs to the transformer. The best model outperforms previous RNN-based state-of-the-arts in terms of standard NLG metrics (BLEU, ROUGE, METEOR) and of whether a standard QA model can correctly answer the generated questions. While our model is simple, our model achieves state-of-the-art performance at both sentence-level and paragraph-level input and provides strong baselines for future research.

Acknowledgments

This work is supported in part by the Ministry of Science and Technology, Taiwan, under grant No: 107-2221-E-005-064-MY2.

References

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the ninth workshop on statistical machine translation*, pages 376–380.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep

bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. *arXiv preprint arXiv:1705.00106*.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 866–874.

Michael Heilman and Noah A Smith. 2010. 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*, pages 609–617. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Igor Labutov, Sumit Basu, and Lucy Vanderwende. 2015. Deep questions without deep understanding. 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)*, volume 1, pages 889–898.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Karen Mazidi and Rodney D Nielsen. 2014. Linguistic considerations in automatic question generation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 321–326.

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 annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Vasile Rus, Brendan Wyse, Paul Piwek, Mihai Lintean, Svetlana Stoyanchev, and Cristian Moldovan. 2010. The first question generation shared task evaluation challenge.

Shikhar Sharma, Layla El Asri, Hannes Schulz, and Jeremie Zumer. 2017. Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation. *arXiv preprint arXiv:1706.09799*.

Linfeng Song, Zhiguo Wang, and Wael Hamza. 2017. A unified query-based generative model for question generation and question answering. *arXiv preprint arXiv:1709.01058*.

Sandeep Subramanian, Tong Wang, Xingdi Yuan, Saizheng Zhang, Yoshua Bengio, and Adam Trischler. 2017. Neural models for key phrase detection and question generation. *arXiv preprint arXiv:1706.04560*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Duyu Tang, Nan Duan, Tao Qin, Zhao Yan, and Ming Zhou. 2017. Question answering and question generation as dual tasks. *arXiv preprint arXiv:1706.02027*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Tong Wang, Xingdi Yuan, and Adam Trischler. 2017. A joint model for question answering and question generation. *arXiv preprint arXiv:1706.01450*.

Xuchen Yao, Gosse Bouma, and Yi Zhang. 2012. Semantics-based question generation and implementation. *Dialogue & Discourse*, 3(2):11–42.

Xingdi Yuan, Tong Wang, Caglar Gulcehre, Alessandro Sordoni, Philip Bachman, Sandeep Subramanian, Saizheng Zhang, and Adam Trischler. 2017. Machine comprehension by text-to-text neural question generation. *arXiv preprint arXiv:1705.02012*.

Yao Zhao, Xiaochuan Ni, Yuanyuan Ding, and Qifa Ke. 2018. 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*, pages 3901–3910.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017. Neural question generation from text: A preliminary study. In *National CCF Conference on Natural Language Processing and Chinese Computing*, pages 662–671. Springer.