Learning to Generate Questions by Learning to Recover Answer-containing Sentences

Seohyun Back[12*], **Akhil Kedia**[1*], **Sai Chetan Chinthakindi**[1], **Haejun Lee**[1] and **Jaegul Choo**[2]
[1]Samsung Research, Seoul, South Korea [2]KAIST, Daejeon, South Korea
{scv.back,sai.chetan,akhil.kedia,haejun82.lee}@samsung.com
jchoo@kaist.ac.kr

Abstract

To train a question answering model based on machine reading comprehension (MRC), significant effort is required to prepare annotated training data composed of questions and their answers from contexts. Recent research has focused on synthetically generating a question from a given context and an annotated (or generated) answer by training an additional generative model to augment the training data. In light of this research direction, we propose a novel pre-training approach that learns to generate contextually rich questions, by recovering answer-containing sentences. We evaluate our method against existing ones in terms of the quality of generated questions, and fine-tuned MRC model accuracy after training on the data synthetically generated by our method. We consistently improve the question generation capability of existing models such as T5 and UniLM, and achieve state-of-the-art results on MS MARCO and NewsQA, and comparable results to the state-of-the-art on SQuAD. Additionally, the data synthetically generated by our approach is beneficial for boosting up the downstream MRC accuracy across a wide range of datasets, such as SQuAD-v1.1, v2.0, KorQuAD and BioASQ, without any modification to the existing MRC models. Furthermore, our method shines especially when a limited amount of pre-training or downstream MRC data is given.

1 Introduction

Machine reading comprehension (MRC), which finds the answer to a given question from its accompanying paragraphs (called context), is an essential task in natural language processing. With the release of high-quality human-annotated datasets for the task, such as SQuAD-v1.1 (Rajpurkar et al., 2016), -v2.0 (Rajpurkar et al., 2018), and KorQuAD (Lim et al., 2019), researchers have proposed MRC models even surpassing human scores. These datasets commonly involve finding a snippet within a context as an answer to a given question.

However, these datasets require significant amount of human effort to create questions and their relevant answers from given contexts. Often the size of the annotated data is relatively small compared to that of data used in other self-supervised tasks such as language modeling, limiting the accuracy.

To overcome this issue, researchers have studied models for generating synthetic questions from a given context along with annotated (or generated) answers on large corpora such as Wikipedia. Golub et al. (2017) suggested a two-stage network of generating question-answer pairs which first chooses answers conditioned on the paragraph and then generates a question conditioned on the chosen answer. Dong et al. (2019) showed that pre-training on unified language modeling from large corpora including Wikipedia improves the question generation capability. Alberti et al. (2019) introduced a self-supervised pre-training technique for question generation via the next-sentence generation task.

However, self-supervised pre-training techniques such as language modeling or next sentence generation are not specifically conditioned on the candidate answer and instead treat it like any other phrase, despite the candidate answer being a strong conditional restriction for the question generation task. Also, not all sentences from a paragraph may be relevant to the questions or answers, so the task of their generation may not be an ideal candidate as a pre-training method for question generation tasks.

To address these issues, we propose a novel training method called Answer-containing Sentence Generation (ASGen) for a question generator. ASGen is composed of two steps: (1) predicting

*These authors contributed equally.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1516–1529

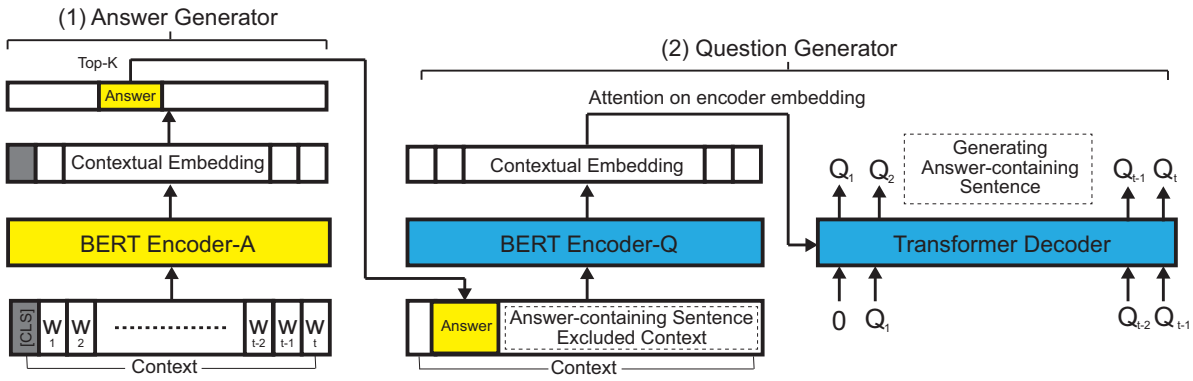


Figure 1: Architecture of a simple generative model, BertGen. When applying our training method "ASGen" to the model, the question generator takes as input the answer and the context with the answer-containing sentence removed and generates the missing answer-containing sentence.

"answer-like" candidate answers in a given context and (2) pre-training the question generator on the answer-containing sentence generation task. We evaluate our method against existing ones in terms of the generated question quality as well as the fine-tuned MRC model accuracy after training on the data synthetically generated by our method.

Experimental results demonstrate that our approach consistently improves the question generation quality of existing models such as T5 (Raffel et al., 2020) and UniLM (Dong et al., 2019), and shows state-of-the-art results on MS MARCO (Nguyen et al., 2016), NewsQA (Trischler et al., 2017), as well as comparable results to the state-of-the-art on SQuAD. Additionally, we demonstrate that the synthetically generated data by our approach can boost downstream MRC accuracy across a wide range of datasets, such as SQuAD-v1.1, v2.0, KorQuAD and BioASQ (Tsatsaronis et al., 2015) without any modification to the existing MRC models. Furthermore, our experiments highlight that our method shines especially when a limited amount of training data is given, in terms of both pre-training and downstream MRC data.

2 Proposed Method

This section discusses our proposed training method called Answer-containing Sentence Generation (ASGen). While ASGen can be applied to any generative model, we use a simple Transformer (Vaswani et al., 2017) based generative model as our baseline, which we call BertGen. First, we will describe how the BertGen model generates synthetic questions and answers from a context. Next, we will explain the details of candidate answer prediction and how we pre-trained the question generator in BertGen based on them. BertGen encodes given paragraphs with two networks, the answer generator and the question generator.

Answer Generator. To make the contextual embeddings and to predict answer spans for a given context without the question, we utilize a BERT (Devlin et al., 2019) encoder (Fig. 1-(1), BERT Encoder-A). We select top K candidate answer spans from the context by sorting with confidence score of span prediction. We use the K selected answer spans as input to the question generator.

Question Generator. Next, we generate a question conditioned on each answer predicted from the answer generator. Specifically, we give as input to a BERT encoder the context and an indicator for the answer span location in the context (Fig. 1-(2), BERT Encoder-Q). Next, a Transformer decoder generates the question word-by-word based on the encoded representation of the context and the answer span. When pre-training the question generator on an answer-containing sentence generation task, we exclude the answer-containing sentence from the original context and train the model to generate the excluded sentence given the modified context and the answer span as input.

Finally, we generate synthetic questions and answers from a large corpus, e.g., all the paragraphs in Wikipedia. After generating this data, we train the MRC model on the generated data in the first phase and then fine-tune on the downstream MRC dataset (e.g., SQuAD) in the second phase. In this paper, we use BERT as the default MRC model, since BERT or its variants achieve state-of-the-art performance across numerous MRC tasks.

2.1 Candidate Answer Prediction

In question generation, it is important to determine which part of a given context can be a suitable answer for generating questions. To this end, we predict candidate answer span in the given context $W = \{\mathbf{w}_t\}_{t=0}^{T}$ to obtain a more appropriate set of "answer-like" phrases. To calculate the score s_i for start index i of a predicted answer span, we compute the dot product of the encoder output with a trainable vector $\mathbf{v}_s$. For each start index i, we calculate the span end index score $e_{i,j}$ for index j in a similar manner with a trainable vector $\mathbf{v}_e$, i.e.,

$$\begin{aligned}\{\mathbf{w}_t^{enc}\}_{t=0}^{T} &= \text{BERT Encoder-A}(W),\\ s_i &= \mathbf{v}_s \circ \mathbf{w}_i^{enc},\\ e_{i,j} &= \mathbf{v}_e \circ f_s(\mathbf{w}_j^{enc} \oplus \mathbf{w}_i^{enc}),\end{aligned}$$

where T is the number of word tokens in the context and f_s represents a fully connected layer with hidden dimension H and $\oplus$ indicates the concatenation operation. Token at $t = 0$ is "[CLS]". For training, we use cross-entropy loss on the s_i, $e_{i,j}$ with ground truth start, end of the answer span for each token.

During inference, we choose top K answer spans with the highest score summation of start index score and end index score, i.e.,

$$\begin{aligned}A^{span} &= \{(i,j) \mid 1 \le i \le T \text{ and } i \le j \le T\},\\ a_k &= \max(\{a \mid \#\{(i,j) \mid s_i + e_{i,j} \ge a\} = K\}),\\ A_k^{span} &= \{(i,j) \mid s_i + e_{i,j} \ge a_k\}.\end{aligned}$$

Selected answer A_k^{span} is then given to the question generator as input in the form of an indication of the answer span location in the given context.

2.2 Pre-training Question Generator

In order to generate questions conditioned on different answers that may arise in a context, we generate a question for each of the K answers. Alberti et al. (2019) proposed a pre-training method for this generative model using the self-supervised task of generating the next-sentence. We identify several issues with this approach. This technique is not specifically conditioned on the answer, despite the answer being a strong condition for the question generation task. Also, not all sentences from a paragraph may be relevant to the questions or answers from within that paragraph, so their generation is not an ideal candidate for pre-training question generation model.

To address these issues, we modify the context to exclude the sentence containing the previously generated answer and pre-train the question generation model on the task of generating this excluded answer-containing sentence, conditioned on the answer and the modified context.

Specifically, we exclude answer-containing sentence S^{ans} while retaining the answer, modifying the original context D to D^{ans} as

$$\begin{aligned}S^{start} &= \{p \mid \text{sentence start index} = p\} \cup \{T\},\\ S^{ans} &= \{(p_s, p_e, i, j) \mid p_s = \max(\{p \le i\}),\\ p_e &= \min(\{p \ge j\}), p \in S^{start}, (i,j) \in A_k^{span}\},\\ D^{ans} &= [D_{[:p_s]}; D_{[i:j]}; D_{[p_e:]}], (p_s, p_e, i, j) \in S^{ans},\end{aligned}$$

Note that we change S^{ans} to not exclude the answer-containing sentence for fine-tuning the question generator, i.e.,

$$S^{ans} = \{(p_s, p_e, i, j) | p_s = i, p_e = j\}.$$

In BertGen, we pass the previously generated answer to the generation model in the form of an additional position encoding M^{ans} that indicates the answer location within the context, i.e.,

$$M^{ans} = [\mathbf{m}_0 * p_s; \mathbf{m}_1 * (j - i); \mathbf{m}_0 * (T - p_e)],$$

where $\mathbf{m}_0$ and $\mathbf{m}_1$ indicate trainable vectors corresponding to encoding id 0 and 1, respectively. That is, we assign the encoding id for each word in the context as 0 and each word in the answer as 1. $A * B$ indicates the operation of stacking vector A for B many times.

Next, we generate answer-containing sentence output words' probability $W^o = \{\mathbf{w}_y^o\}_{y=1}^{Y}$ as

$$\begin{aligned}C^{enc} &= \text{BERT Encoder-Q}(D^{ans}, M^{ans}),\\ \mathbf{w}_y^g &= \text{Transformer Decoder}(\{\mathbf{w}_i^o\}_{i=0}^{y-1}, C^{enc}),\\ \{\mathbf{w}_y^o\}_{y=1}^{Y} &= \{\text{Softmax}(\mathbf{w}_y^g E)\}_{y=1}^{Y},\end{aligned}$$

where C^{enc} is encoded representation of the context and $E \in \mathbb{R}^{d \times V}$ represents a word embedding matrix with vocabulary size V shared between the BERT Encoder-Q and the decoder. Note that $\mathbf{w}_0^o$ is a zero vector for starting the decoding.

Finally, we calculate the loss of the generated words using the cross-entropy loss as

$$\mathbb{L} = -\left(\sum_{y=1}^{Y}\sum_{v=1}^{V} \mathbf{z}_{y,v} \log(\mathbf{w}_{y,v}^o)\right)/Y,$$

where $\mathbf{z}$ indicates a ground-truth one-hot vector of the answer-containing sentence word. Note that $\mathbf{z}$ is the question word in the case of fine-tuning.

In this manner, we pre-train the question generation model using a task similar to the final task of conditionally generating the question from a given answer and a context.

3 Experimental Setup

Pre-training Dataset. To build the dataset for answer-containing sentence generation tasks (AS-Gen) and the synthetic MRC data for pre-training the downstream MRC models, we collect all paragraphs from the entire English Wikipedia dump and synthetically generate questions and answers on these paragraphs. Note that we removed all passages from Wikipedia overlapping with SQuAD dataset (Rajpurkar et al., 2016). We apply filtering and clean-up steps that are detailed in the appendix.

Using BertGen, we extract answers from each given paragraph, and then generate questions for each answer-paragraph pair. Finally, we obtain 43M triples of question-answer-paragraph for the synthetic data. For pre-training on answer-containing sentence generation, we sample 25M answer-paragraph pairs (Full-Wiki) from the final Wikipedia dataset to avoid extremely short contexts less than 500 characters. For ablation studies on pre-training approaches, we sample 2.5M pairs (Small-Wiki)[1] from Full-Wiki and split 25K pairs (Test-Wiki) to evaluate the pre-training method.

Benchmark Datasets. In most MRC datasets, a question and a context are represented as a sequence of words, and the answer span (indices of start and end words) is annotated from the context words based on the question. Among these datasets, we choose SQuAD as the primary benchmark dataset for question generation, since it is the most popular human-annotated MRC dataset. For fair comparison with existing question generation methods, we use the same splits of SQuAD-v1.1, as previously done in Du et al. (2017), Kim et al. (2019), and Dong et al. (2019). We refer to this dataset as Split1. This split has 77K/10K/10K samples for train/dev/test sets. We also evaluate on the reversed dev-test split, referred to as Split2.[2] Additionally, we test our question generation on MS MARCO (Nguyen et al., 2016) and NewsQA (Trischler et al., 2017) to evaluate the generalization of our method to other datasets. In the case of MS MARCO, questions are collected from real user query logs in Bing. For these datasets, we follow pre-processing of Tuan et al. (2020), sampling a subset of original data where the answers are sub-spans of their corresponding paragraphs to obtain train/dev/test sets with 51K/6K/7K samples for MS MARCO and 76K/4K/4K samples for NewsQA. We also conduct experiments on question generation with Natural Questions (Kwiatkowski et al., 2019) and BioASQ (Tsatsaronis et al., 2015). We calculate BLEU-4, METEOR, and ROUGE-L with the script from Du et al. (2017).

To evaluate the effectiveness of generated synthetic MRC data, we test the fine-tuned MRC model on the downstream MRC dataset after training on the generated synthetic data. We calculate the EM/F1 score of the MRC model on SQuAD-v1.1 and v2.0 development set. We also evaluate on the test set of KorQuAD, a Korean dataset created with the same procedure as SQuAD-v1.1.

Implementation Details. For all experiments and models, we use all official original hyperparameters unless otherwise stated below. For BertGen model, we use pre-trained BERT (Base and Large) as encoder and 12 stacked layers of Transformer as decoder. For large version of the model, we use 24 layers of the encoder and the decoder with 737M parameters. For answer prediction, we select top-5 ($K = 5$) for the answer spans. For the generation of unanswerable questions in SQuAD-v2.0, we separate unanswerable and answerable cases and then train separate generation models. For all BertGen models, we pre-train the question generator for 5 epochs on Wikipedia and fine-tune it for 30 epochs on MRC dataset with batch size of 32. For other question generation models, we pre-train for 1 epoch on Wikipedia. For UniLM and T5, the input is formulated as sequence-to-sequence, the first input segment is the concatenation of context and answer, while the second output segment is a missing answer-containing sentence or a question to be generated. We use all official settings for UniLM, ProphetNet (Qi et al., 2020) and ELECTRA (Clark et al., 2020), and use the official pre-trained weights. The training time depends on the data size and the model complexity. For Zhao et al. (2018), pre-training on Full-Wiki takes 48 hours. Pre-training BertGen on Small-Wiki in Table 3 takes 48 hours with 8 Tesla V100 GPU, resulting

[1] We use the Korean Wikipedia for KorQuAD.

[2] We use the same splits as provided by Du et al. (2017)

Table 1: Comparison with existing question generation methods on the test set of SQuAD Split1 and Split2. Models marked as '*' indicate results we reproduced.

Group	Question Generation Model	Split1			Split2		
		BLEU-4	METEOR	ROUGE-L	BLEU-4	METEOR	ROUGE-L
	Du et al. (2017)	12.3	16.6	39.8	-	-	-
	ASs2s (Kim et al., 2019)	16.2	19.9	44.0	-	-	-
Applying to other methods	Zhao et al. (2018)*	13.0	18.2	41.2	15.1	19.5	43.4
	Zhao et al. (2018)* + ASGen	14.2	19.4	42.8	16.4	20.6	44.7
	T5(Small)*	15.6	23.3	37.1	18.8	25.2	40.5
	T5(Small)* + ASGen	17.0	24.2	38.9	19.6	26.1	41.9
	T5(Large)*	18.9	26.2	41.8	21.7	27.4	44.5
	T5(Large)* + ASGen	19.4	**26.7**	42.0	22.0	**27.8**	45.1
	UniLM (Dong et al., 2019)	22.1	25.1	51.1	23.8	25.6	52.0
	UniLM + ASGen	23.7	25.9	52.3	25.3	26.7	53.3
	ProphetNet (Qi et al., 2020)	23.9	26.6	52.3	25.8	27.5	53.7
	ProphetNet + ASGen	**24.4**	**26.7**	**52.8**	**26.1**	27.6	**53.9**
	BertGen (Large) + ASGen	22.8	25.3	51.2	24.6	25.8	53.0

Table 2: Comparison with existing question generation methods on MS MARCO and NewsQA. We also test our method on Natural Questions. BL-4, MTR, RG-L indicate BLEU-4, METEOR, ROUGE-L.

MS MARCO (test set)	BL-4	MTR	RG-L
Zhao et al. (2018)	17.2	-	-
Tuan et al. (2020)	18.3	19.4	42.8
Ma et al. (2020)	20.5	24.7	49.9
BertGen (Large) + ASGen	**22.9**	**26.7**	**51.8**
NewsQA (test set)	BL-4	MTR	RG-L
Zhou et al. (2017)	9.9	16.7	42.3
Liu et al. (2019)	11.1	17.4	43.2
Tuan et al. (2020)	12.4	**19.0**	44.1
BertGen (Large) + ASGen	**13.8**	18.6	**44.5**
Natural Questions (dev set)	BL-4	MTR	RG-L
BertGen (Large)	31.5	30.4	60.2
BertGen (Large) + ASGen	**35.3**	**32.9**	**61.3**

Table 3: Ablation of pre-training methods, i.e., pre-training on NS, ASGen, and ASGen without conditioning on a given answer (w/o A), on the test set of SQuAD splits and on Test-Wiki.

Pre-train on Small-Wiki	Wiki	Split1	Split2
BertGen (w/o pre-train)	-	15.0	17.1
BertGen+NS	1.4	19.0	20.2
BertGen+ASGen w/o A	**5.2**	19.9	21.0
BertGen+ASGen	**5.2**	**20.1**	**21.4**
Pre-train on Full-Wiki	Wiki	Split1	Split2
BertGen+NS	3.4	20.6	22.6
BertGen+ASGen	8.2	22.2	24.2
BertGen(Large)+ASGen	**8.3**	**22.8**	**24.6**

Table 4: Average of 10 human evaluation scores over random samples from SQuAD. Each column indicates Syntax (ST), Semantics (SM), Context-Relevance (CR) and Answer-Relevance (AR) in the range 1 to 5.

Model	ST	SM	CR	AR
BertGen	4.04	3.93	4.20	3.25
BertGen+NS	4.60	4.54	4.49	3.63
BertGen+ASGen	**4.71**	**4.69**	**4.74**	**4.14**
UniLM	4.25	4.31	4.54	4.06
UniLM+ASGen	**4.71**	**4.79**	**4.70**	**4.17**

in 5.1, 4.3 BLEU-4 improvement on Split1, Split2 respectively. The pre-training for BertGen (Large) with Full-Wiki takes 1,224 hours and fine-tuning takes 72 hours. Mecab (Kudo, 2006) is used for Korean tokenizer.

Comparison of the Pre-training Method. We compare ASGen with a method from Alberti et al. (2019), which is pre-training on next-sentence generation task (NS), and with a method from Golub et al. (2017), which only trains the generative model on the final MRC dataset. We reproduced these methods on BertGen as described in their original work and evaluate question generation scores on the SQuAD splits as well as corresponding sentence generation scores on Test-Wiki.

Comparison of Downstream Results. To check the effectiveness of our method on downstream MRC tasks, we evaluate our generated synthetic data on SQuAD-v1.1, v2.0, and KorQuAD by training MRC models (BERT, BERT+CLKT and ELECTRA) on generated data followed by fine-tuning on the train set for each dataset. The structure of BERT+CLKT model is the same as that of original BERT except that the model is pre-trained for the Korean language. Due to the absence of common pre-trained BERT for Korean, we used this model.

4 Experimental Results

4.1 Question Generation Performance

Comparison to Existing Methods. To evaluate ASGen, we fine-tune the question generation models on both SQuAD splits, after pre-training on answer-containing sentence generation task. As shown in Table 1, 'BertGen (Large) + ASGen' and 'UniLM + ASGen' outperforms UniLM on both splits. Moreover, ASGen consistently improves the performance when applied to other question generation models such as Zhao et al. (2018), T5 (Small and Large), and UniLM across all metrics for both splits. In particular, applying ASGen on UniLM further improves its question generation capability, achieving BLEU-4, METEOR, and ROUGE-L as 23.7, 25.9, 52.2, and 25.3, 26.7, 53.3 on both splits, respectively. We reproduce Zhao et al. (2018) and T5, and use the official code of UniLM with no architecture or parameter changes.

Additionally, as shown in Table 2, 'BertGen (Large) + ASGen' outperforms all existing models on all scores on both MS MARCO and NewsQA, except for comparable METEOR scores in NewsQA. Our method also shows improvement on Natural Questions (Kwiatkowski et al., 2019) (short answer) dataset, where questions are collected from real user query logs on Google.

Ablation Study of Pre-training Task. We also compare the BLEU-4 scores between various pre-training tasks to show the effectiveness of ASGen. As shown in Table 3, ASGen outperforms NS in the recreation score of sentence on Test-Wiki, e.g. 5.2 vs. 1.4 in Small-Wiki and 8.2 vs. 3.4 in Full-Wiki. ASGen outperforms NS in question generation, e.g. 22.2 vs. 20.6 and 24.2 vs. 22.6 in the two splits, respectively. We also observe that conditioning on a given answer improves ASGen, e.g. 20.1 vs. 19.9 in Split1 and 21.4 vs. 21.0 in Split2.

Human Evaluation. As Sultan et al. (2020) mentioned in their paper, accuracy-based measurements such as BLEU-4, METEOR and ROUGE-L may not be adequate to test the diversity of a question. Due to this, we also judge the quality of questions by human evaluation involving 10 evaluators over metrics such as syntax, validation of semantics, question to context relevance and question to answer relevance on 50 randomly chosen samples on SQuAD-v1.1 dev set. As shown in Table 4, applying ASGen consistently improves the human evaluation scores.

Table 5: Effectiveness of synthetic data for MRC model on SQuAD (SQD) and KorQuAD (KQD).

MRC model	Dev-SQDv1.1 EM	Dev-SQDv1.1 F1	Dev-SQDv2.0 EM	Dev-SQDv2.0 F1
BERT (Large)	83.9	90.9	78.8	81.8
+synthetic data	**86.3**	**92.7**	**84.5**	**87.4**
BERT (WWM)	86.5	92.8	83.1	85.9
+synthetic data	**87.4**	**93.5**	**85.5**	**88.4**
ELECTRA (Large)	-	-	87.4	90.2
+synthetic data	-	-	**88.2**	**91.3**
MRC model	Dev-KQD EM	Dev-KQD F1	Test-KQD EM	Test-KQD F1
BERT+CLKT	87.1	94.5	86.2	94.1
+synthetic data	**87.8**	**95.0**	**86.7**	**94.6**

Table 6: Comparison of downstream EM/F1 scores using BERT(Large) MRC model with the synthetic data from different pre-training methods.

Synthetic Data	Dev-v1.1 EM	Dev-v1.1 F1	Dev-v2.0 EM	Dev-v2.0 F1
BertGen (w/o pre-train)	85.1	91.4	80.9	83.9
BertGen+NS	85.6	92.3	81.5	85.8
BertGen+ASGen	**86.3**	**92.7**	**84.5**	**87.4**

4.2 Downstream MRC Task Performance

To show the effectiveness of the generated synthetic data, we train MRC models on generated data, before fine-tuning on the downstream data. As shown in Table 5, the synthetic data generated by 'BertGen (Large) + ASGen' consistently improves the performance of BERT (Large, WWM) by a significant margin. Pre-training BERT on synthetic data improves F1 scores by 1.8 on SQuAD-v1.1 and 5.6 on SQuAD-v2.0 for BERT (Large), and 0.7 on SQuAD-v1.1 and 2.5 on SQuAD-v2.0 for BERT (WWM). Synthetic data also improves ELECTRA performance on SQuAD-v2.0, and BERT+CLKT performance on KorQuAD.

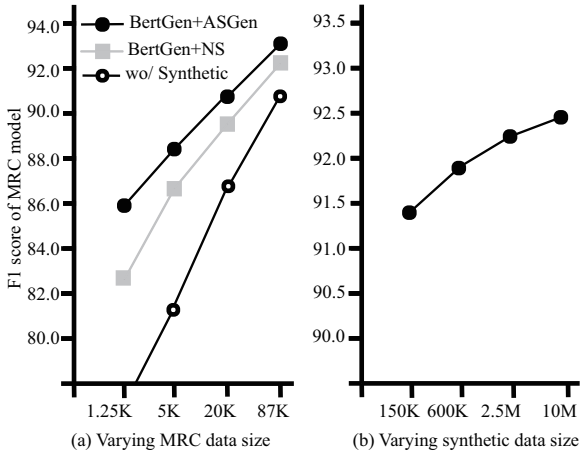
Also, to show improvement due to our pre-training method in the downstream MRC task, we compare the EM/F1 scores of BERT (Large) models trained on synthetic data generated by different question generation models, 'BertGen', 'BertGen + NS' and 'BertGen + ASGen'. As shown in Table 6, our method outperforms other methods both on SQuAD-v1.1 and SQuAD-v2.0.

4.3 Effects of Training Data Size

Fig. 2 shows the effects of varying amounts of downstream MRC data and synthetic data on F1 scores of BERT (Large). In Fig. 2-(a), where we fix the size of synthetic data as 43M, pre-training with

Table 7: The performance of our method on limited-data domain (BioASQ). Note that the scores of question generation are obtained from BioASQ factoid-type 6b. All experiments used the official code of Yoon et al. (2020).

Question Generation Model		BLEU-4	METEOR	ROUGE-L
BertGen (Large)		6.6	10.0	33.1
BertGen (Large) + ASGen (Full-Wiki)		**12.6**	**17.8**	**40.0**
MRC model	Pre-training Data	Factoid (MRR)	Yes/No (Macro F1)	List-Type (F1)
BERT(Large)	-	34.3	53.8	36.1
BERT(Large)	ASGen (Full-Wiki)	49.2	**81.1**	**39.8**
BioBERT(Large)	PubMed	**52.3**	80.1	38.1



(a) Varying MRC data size (b) Varying synthetic data size

Figure 2: F1 scores of BERT (Large) on SQuAD-v1.1 dev by limiting size of MRC and synthetic data.

'BertGen + ASGen' consistently outperforms 'BertGen + NS' for all sizes of downstream data. While the performance difference is particularly apparent for smaller sizes of downstream data, it persists even on using the entire MRC data (SQuAD-v1.1). In Fig. 2-(b), we also conduct experiments by training BERT (Large) using different amounts of generated synthetic data while keeping the number of pre-training steps constant and using the full size of downstream MRC data. Increasing the amount of synthetic data used consistently improves the accuracy of the MRC model.

4.4 Transfer Learning to Limited Domain

We also conduct experiments on BioASQ (Tsatsaronis et al., 2015) dataset to show the effectiveness of our model in limited-data domains having less annotated data. As shown in Table 7, ASGen improves the question generation scores by 6.0 BLEU-4, 7.8 METEOR and 6.9 ROUGE-L on BioASQ factoid-type 6b. Moreover, using 'Full-Wiki' data enhances the performance of BERT(Large) by a large margin and outperforms BioBERT (Lee et al., 2019a), by 0.95 Macro F1 (Yes/No) and 1.63 F1 (List). Note that BioBERT is specifically pre-trained on a medical corpus (PubMed) whereas we use a generic Wikipedia corpus ('Full-Wiki'), with our generation models fine-tuned on SQuAD.

Table 8: Manual categorization of the reasoning type for generated answerable questions. Note that each example can be assigned to multiple types.

Reasoning Type	BertGen +ASGen	SQuAD v1.1
Lexical Variation (Synonymy)	40.7%	33.3%
Lexical Variation (World Knowledge)	4.0%	9.1%
Syntactic Variation	53.3%	64.1%
Multi Sentence Reasoning	21.3%	13.6%
Ambiguous/Unanswerable	4.0%	6.1%

4.5 Qualitative Analysis of Generation

Comparison of Sample Questions. We qualitatively compare the generated questions after pre-training BertGen with NS and ASGen to demonstrate the effectiveness of our method. For the correct answer "49.6%" as shown in the first sample in Table 9, the word "Fresno", which is critical to make the question specific, is omitted by NS, while ASGen's question does not suffer from this issue. Note that the word "Fresno" occurs in the answer-containing sentence. This issue also occurs in the second sample, where NS uses the word "available" rather than relevant words from the answer-containing sentence, but ASGen uses many of these words such as "most" and "popular" to generate contextually rich questions. Also, the question from NS is about "two" libraries, while the answer is about "three" libraries, showing the lack of sufficient conditioning on the answer. Similarly, the third example also shows that ASGen generates more contextual questions than NS by including the exact subject "TARDIS" based on the corresponding answer. Based on these observations and from the score improvements in Table 3, we conjecture that ASGen leads the question generation model to better condition on the answer and to

Table 9: Examples of questions generated on SQuAD-v1.1 development set. We compare generated questions from 'BertGen + ASGen' with 'BertGen + NS'. Colored Text indicates given answers.

Context	(omit) ... The population density was 4,404.5 people per square mile. (1,700.6km). The racial makeup of Fresno was 245,306 (49.6%) White, 40,960 (8.3%) ... (omit)
BertGen + NS	What percent of the population is White?
BertGen + ASGen	What percentage of the Fresno population is White?
Context	(omit) ... in the world. Cabot Science Library, Lamont Library, and Widener Library are three of the most popular libraries for undergraduates to use ... (omit)
BertGen + NS	Which two libraries are available for undergraduates to use?
BertGen + ASGen	What are the three most popular libraries for undergraduates?
Context	(omit) ... in a stolen Mark I Type TARDIS "Time and Relative Dimension in Space" time machine which allows him to travel across time and space. ... (omit)
BertGen + NS	What does the doctor refer to?
BertGen + ASGen	What does the TARDIS stand for?

Table 10: Manual categorization of the reasoning type for unanswerable questions.

Reasoning Type	BertGen +ASGen	SQuAD v2.0
Negation	8.0%	9.0%
Antonym	14.7%	20.0%
Entity Swap	36.0%	21.0%
Mutual Exclusion	9.3%	15.0%
Impossible Condition	7.3%	4.0%
Other Neutral	19.3%	24.0%
Answerable	5.3%	7.0%

generate more contextualized questions than NS.

Categorization of Reasoning Type. We manually categorized the reasoning type of 150 randomly sampled generated questions on Wikipedia for both answerable and unanswerable questions. The results Table 8 and Table 10 show that generated questions using ASGen often require multi-hop or other non-trivial reasoning. We follow the same categorization as done by Rajpurkar et al. (2016, 2018).

5 Related Work

Question Generation. Researchers have actively studied question generation for various purposes, including for data augmentation in question answering. Du et al. (2017) proposed an attention-based model for question generation by encoding sentence-level as well as paragraph-level information. Zhao et al. (2018) utilized a gated self-attention encoder with a max-out unit to handle long paragraphs. Song et al. (2018) introduced a query-based generative model to jointly solve question generation and answering tasks. Kim et al. (2019) separately encoded the answer and the rest of the paragraph for question generation. Ma et al. (2020) suggested sentence-level semantic matching and answer-position-aware question generation. Tuan et al. (2020) show that incorporating interactions across multiple sentences enhances question generation performance. Our approach can further improve the question generation quality of these methods by pre-training them with the answer-containing sentence generation task.

Transfer Learning. Pre-training methods are popular in natural language processing for learning contextualized word representations. Open-GPT (Radford et al., 2018), BERT (Devlin et al., 2019), XLNet (Yang et al., 2019), PEGASUS (Zhang et al., 2019), ERNIE-GEN (Xiao et al., 2020), UniLM (Dong et al., 2019), UniLMv2 (Bao et al., 2020), T5 (Raffel et al., 2020) and BART (Lewis et al., 2020) utilize Transformer (Vaswani et al., 2017) to learn different types of language models on a large dataset followed by fine-tuning on a downstream task. These pre-training approaches tend to be very generic, while our approach is a more appropriate pre-training method focused on the specific task of question generation. Lee et al. (2019b) suggested a pre-training method for information retrieval called Inverse Cloze Task. Unlike this method, our pre-training task for the question generator is strongly conditioned on the answer and focuses on generating missing answer-containing sentence in the context to learn better representations more suitable to the question generation task.

Synthetic Data Generation. Subramanian et al. (2018) show that neural models generate better candidate answers from a given paragraph than using off-the-shelf tools or selecting named entities and noun phrases. Yang et al. (2017) introduced a training method for the MRC model by combining synthetic data and human-annotated data. Similar to

our method, Golub et al. (2017) proposed to generate questions conditioned on generated answers by separating the answer generation and the question generation. Unlike this paper, they do not pre-train their question generator on the answer-containing sentences. Dong et al. (2019) also show that utilizing synthetic data boosts the performance of MRC models. Inspired by these previous studies, we propose a newly designed pre-training technique that improves capability of question generation models.

6 Conclusions

We propose a novel pre-training method called AS-Gen to learn generating contextually rich questions better conditioned on the answers. Our approach improves question generation ability of existing methods, achieves new state-of-the-art results on MS MARCO and NewsQA, and the synthetic data increases downstream MRC accuracy across a wide range of datasets without any modification to the existing MRC models.

Acknowledgements

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No. 2019-0-00075, Artificial Intelligence Graduate School Program(KAIST)), the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. NRF-2019R1A2C4070420), and Institute for Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No. 2020-0-00368, A Neural-Symbolic Model for Knowledge Acquisition and Inference Techniques).

References

Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. 2019. Synthetic QA corpora generation with roundtrip consistency. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 6168–6173. Association for Computational Linguistics.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: an automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization@ACL 2005, Ann Arbor, Michigan, USA, June 29, 2005*, pages 65–72. Association for Computational Linguistics.

Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Songhao Piao, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2020. Unilmv2: Pseudo-masked language models for unified language model pre-training. *CoRR*, abs/2002.12804.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Bhuwan Dhingra, Kathryn Mazaitis, and William W. Cohen. 2017. Quasar: Datasets for question answering by search and reading. *CoRR*, abs/1707.03904.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 13042–13054.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1342–1352. Association for Computational Linguistics.

David Golub, Po-Sen Huang, Xiaodong He, and Li Deng. 2017. Two-stage synthesis networks for transfer learning in machine comprehension. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 835–844. Association for Computational Linguistics.

Yanghoon Kim, Hwanhee Lee, Joongbo Shin, and Kyomin Jung. 2019. Improving neural question generation using answer separation. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The*

Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019, pages 6602–6609. AAAI Press.

Taku Kudo. 2006. Mecab: Yet another part-of-speech and morphological analyzer. *http://mecab. sourceforge. jp*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Trans. Assoc. Comput. Linguistics*, 7:452–466.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019a. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019b. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, Florence, Italy. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Seungyoung Lim, Myungji Kim, and Jooyoul Lee. 2019. Korquad1.0: Korean QA dataset for machine reading comprehension. *CoRR*, abs/1909.07005.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81.

Bang Liu, Mingjun Zhao, Di Niu, Kunfeng Lai, Yancheng He, Haojie Wei, and Yu Xu. 2019. Learning to generate questions by learningwhat not to generate. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 1106–1118. ACM.

Xiyao Ma, Qile Zhu, Yanlin Zhou, and Xiaolin Li. 2020. Improving question generation with sentence-level semantic matching and answer position inferring. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8464–8471. AAAI Press.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016*, volume 1773 of *CEUR Workshop Proceedings*. CEUR-WS.org.

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 of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318. ACL.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 2401–2410.

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/researchcovers/languageunsupervised/language understanding paper.pdf*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 784–789. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392. The Association for Computational Linguistics.

Linfeng Song, Zhiguo Wang, and Wael Hamza. 2018. A unified query-based generative model for question generation and question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Sandeep Subramanian, Tong Wang, Xingdi Yuan, Saizheng Zhang, Adam Trischler, and Yoshua Bengio. 2018. Neural models for key phrase extraction and question generation. In *Proceedings of the Workshop on Machine Reading for Question Answering@ACL 2018, Melbourne, Australia, July 19, 2018*, pages 78–88. Association for Computational Linguistics.

Md Arafat Sultan, Shubham Chandel, Ramón Fernandez Astudillo, and Vittorio Castelli. 2020. On the importance of diversity in question generation for QA. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5651–5656, Online. Association for Computational Linguistics.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. Newsqa: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP, Rep4NLP@ACL 2017, Vancouver, Canada, August 3, 2017*, pages 191–200. Association for Computational Linguistics.

George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, Yannis Almirantis, John Pavlopoulos, Nicolas Baskiotis, Patrick Gallinari, Thierry Artieres, Axel Ngonga, Norman Heino, Eric Gaussier, Liliana Barrio-Alvers, Michael Schroeder, Ion Androutsopoulos, and Georgios Paliouras. 2015. An overview of the bioasq large-scale biomedical semantic indexing and question answering competition. *BMC Bioinformatics*, 16:138.

Luu Anh Tuan, Darsh J. Shah, and Regina Barzilay. 2020. Capturing greater context for question generation. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 9065–9072. AAAI Press.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.

Dongling Xiao, Han Zhang, Yu-Kun Li, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. 2020. ERNIE-GEN: an enhanced multi-flow pre-training and fine-tuning framework for natural language generation. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3997–4003. ijcai.org.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 5754–5764.

Zhilin Yang, Junjie Hu, Ruslan Salakhutdinov, and William Cohen. 2017. Semi-supervised QA with generative domain-adaptive nets. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1040–1050, Vancouver, Canada. Association for Computational Linguistics.

Wonjin Yoon, Jinhyuk Lee, Donghyeon Kim, Minbyul Jeong, and Jaewoo Kang. 2020. Pre-trained language model for biomedical question answering. In *Machine Learning and Knowledge Discovery in Databases*, pages 727–740, Cham. Springer International Publishing.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2019. PEGASUS: pre-training with extracted gap-sentences for abstractive summarization. *CoRR*, abs/1912.08777.

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, Brussels, Belgium, October 31 - November 4, 2018*, pages 3901–3910. Association for Computational Linguistics.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017. Neural question generation from text: A preliminary study. In *Natural Language Processing and Chinese Computing - 6th CCF International Conference, NLPCC 2017, Dalian, China, November 8-12, 2017, Proceedings*, volume 10619 of *Lecture Notes in Computer Science*, pages 662–671. Springer.

Supplemental Material (Appendix)

A Question Generation on more Datasets

We also evaluate the question generation model on another data split (Split3) from Zhao et al. (2018). Split3 is obtained by dividing the original development set in SQuAD-v1.1 into two equal halves randomly and choosing one of them as the development set and the other as test set while retaining the train set in SQuAD-v1.1. As shown in Table 11, applying ASGen to the reproduced question generation model from Zhao et al. (2018) improves BLEU-4, METEOR, and ROUGE-L score on Split3 by 1.3, 0.9, and 1.3, respectively.

Table 11: Additional experiments on the effectiveness of ASGen on the test set of SQuAD Split3. Small-Wiki is used to pre-train the models. Models with '*' indicate those results we reproduced.

Model + pre-training method	BL-4	MTR	RG-L
Zhao et al. (2018)	16.8	20.6	44.9
Zhao et al. (2018)*	16.3	20.3	44.5
Zhao et al. (2018)* + ASGen	**17.6**	**21.2**	**45.8**

B Training Electra MRC Model using only Small Generated Synthetic Data

To further study the effect of training data size, we apply our synthetic data to the ELECTRA (Clark et al., 2020) MRC model. In Table 12, we report the mean EM/F1 score on SQuAD 2.0 development set of four runs by using official Electra source code[3] and the pre-trained checkpoint. Pre-training ELECTRA on the generated synthetic data using ASGen improves 0.8 EM and 1.1 F1 score on the downstream MRC dataset, SQuAD-v2.0.

Table 12: Ablation study of applying our method to ELECTRA (Clark et al., 2020) on SQuAD-v2.0 dev set after pre-training on the generated synthetic data using ASGen with Small-Wiki.

MRC model	Synthetic Data	Dev set EM	Dev set F1
ELECTRA (Large)	-	87.4	90.2
	'Small-Wiki'	87.9	90.8
	'Full-Wiki'	**88.2**	**91.3**

C Additional Downstream MRC Task Performance

Additionally to show the effectiveness of the generated synthetic data, we also train MRC models on generated data, before fine-tuning on two downstream datasets, Natural Questions and NewsQA. As shown in Table 13, the synthetic data generated by 'BertGen (Large) + ASGen' consistently improves the F1 score of baseline BERT models.

D Transfer Learning to Other MRC Dataset (QUASAR-T)

To show that our generated data is useful for other MRC datasets, we fine-tune and test the MRC model on QUASAR-T (Dhingra et al., 2017), which is another large-scale MRC dataset, after training on the synthetic data generated from SQuAD-v1.1. In this experiment, we first fine-tune 'BertGen + ASGen' using SQuAD-v1.1, and using synthetic data generated by this model, we train the BERT (Large) MRC model. Afterwards, we fine-tune BERT (Large) for the downstream MRC task using QUASAR-T data. QUASAR-T has two separate datasets, one with short snippets as context, and the other with long paragraphs as context. As shown in Table 14, training with our synthetic data improves the F1 score on the test set by 2.2 and 1.7 for the two cases, respectively.

[3]https://github.com/google-research/electra

Table 13: Effectiveness of synthetic data for MRC model on dev set of Natural Questions and NewsQA.

MRC model	Natural Questions (Short) P	R	F1
BERT (Joint)	**60.09**	46.00	52.10
+synthetic data	59.29	**48.22**	**53.17**
MRC model	**Natural Questions (Long) P**	**R**	**F1**
BERT (Joint)	61.45	**68.61**	64.83
+synthetic data	**63.75**	67.50	**65.56**
MRC model	**NewsQA EM**		**F1**
BERT (Large)	51.96		62.54
+synthetic data	**54.73**		**64.53**

Table 14: EM/F1 scores of the BERT (Large) fine-tuned on QUASAR-T dataset. The used synthetic data is generated from ASGen trained on SQuAD-v1.1 (Full-Wiki).

Synthetic Data	Short(Dev) EM	Short(Dev) F1	Short(Test) EM	Short(Test) F1
-	74.3	78.6	74.1	77.8
Full-Wiki	**76.5**	**80.1**	**76.5**	**80.0**
Synthetic Data	**Long(Dev) EM**	**Long(Dev) F1**	**Long(Test) EM**	**Long(Test) F1**
-	72.1	75.6	72.1	74.8
Full-Wiki	**74.2**	**77.4**	**73.8**	**76.5**

E Details of Wikipedia Preprocessing

To build the answer-containing sentence generation data and the synthetic MRC data for SQuAD (Rajpurkar et al., 2016), we collect all paragraphs from all articles of the entire English Wikipedia dump and generate questions and answers on these paragraphs. We apply extensive filtering and clean-up to only retain the highest-quality paragraphs from Wikipedia, as follows.

To filter out low-quality articles, we remove those with less than 200 cumulative page-views including all re-directions in a two-month period.

In order to calculate the number of page-views, official Wikipedia page-view dumps were used. Of the 5.4M original Wikipedia articles, filtering by page-views leaves 2.8M articles. We also remove those articles with less than 500 characters, as they are often low-quality stub articles, which further removes additional 16% of the articles. We remove all "meta" namespace pages such as talk, disambiguation, user pages, portals, etc. as they often contain irrelevant text or casual conversations between editors. In order to extract clean text from the wiki-markup format of the Wikipedia articles, we remove extraneous entities from the markup including table of contents, headers, footers, links/URLs, image captions, IPA double parentheticals, category tables, math equations, unit conversions, HTML escape codes, section headings, double brace templates such as info-boxes, image galleries, HTML tags, HTML comments, and all tables.

We then split the cleaned text into paragraphs and remove all paragraphs with less than 150 characters or more than 3,500 characters. Paragraphs with the number of characters between 150 to 500 were sub-sampled such that these paragraphs make up 16.5% of the final dataset, as originally done for the SQuAD dataset. Since the majority of the paragraphs in Wikipedia are rather short, out of the 60M paragraphs from the final 2.4M articles, our final Wikipedia dataset contains 8.3M paragraphs. Finally, we generate 43M answer-paragraph pairs from the final Wikipedia dataset with the answer generator of BertGen in this paper.

F Central Tendency and Variation for Human Evaluation

Human evaluation involves 10 evaluators over metrics such as syntax (ST), validation of semantics (SM), question to context relevance (CR) and question to answer relevance (AR) on 50 randomly chosen samples on SQuAD-v1.1 development set. Each score is in the range 1 to 5. Central tendency and variation can be found in Table 15.

G Central Tendency and Variation for the Downstream Tasks

For the EM and F1 scores on downstream SQuAD-v1.1 and v2.0 development set in our main paper, we selected 5 model checkpoints from the same pre-training on the synthetic data in different numbers of training steps. We then fine-tuned each of these models on the final downstream data three times each, chose the best performing model on the development set, and reported its score. Central tendency and variation can be found in Table 16.

Table 15: Central tendency and variation for human evaluation scores. ± is 95% confidence interval.

Model	ST	SM	CR	AR
BertGen	4.04 ±0.18	3.93 ±0.19	4.20 ±0.16	3.25 ±0.22
BertGen + NS	4.60 ±0.12	4.54 ±0.13	4.49 ±0.14	3.63 ±0.22
BertGen + ASGen	**4.71** ±0.10	**4.69** ±0.11	**4.74** ±0.09	**4.14** ±0.18
UniLM	4.25 ±0.16	4.31 ±0.16	4.54 ±0.12	4.06 ±0.19
UniLM + ASGen	**4.71** ±0.11	**4.79** ±0.09	**4.70** ±0.11	**4.17** ±0.18

Table 16: Central tendency and variation for the score of our approach, BertGen(Large) + ASGen, on downstream SQuAD-v1.1 and v2.0 dataset. ± is standard deviation.

MRC model	Dev-v1.1		Dev-v2.0	
	EM	F1	EM	F1
BERT (Large)	86.2 ±0.1	92.7 ±0.1	84.4 ±0.2	87.3 ±0.1
BERT (WWM)	**87.4** ±0.1	**93.4** ±0.1	85.5 ±0.1	88.3 ±0.1
Electra (Large)	- -	- -	**88.4** ±0.3	**91.2** ±0.1

H Details of Generating Unanswerable Questions

The mechanism of generating questions may differ in generating answerable and unanswerable questions. For example, the model could exploit a mismatched phrase to make a question plausible but unanswerable. In order to reflect these characteristics, we train answerable and unanswerable models separately. We first take the BertGen model pre-trained on the ASGen task and then fine-tune this model on the no-answer question generation on SQuAD-v2.0. We infer with this model on the entire Wikipedia to make negative examples for un-answerble synthetic data for pre-training MRC models on SQuAD-v2.0.

I BLEU-4, METEOR, and ROGUE-L

BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005) and ROUGE (Lin, 2004) are widely-used metrics for evaluating the quality of

generated text, where the quality indicates the degree of correspondence between generated text and reference texts. BLEU uses modified precision to compare a generated text against the reference texts. BLEU-4 calculates a weighted score of unigram, bigram, trigram, and 4-gram based matching. METEOR uses harmonic mean between precision and recall of unigrams, but with for recall given more importance than precision. Unlike BLEU, METEOR also tries to match synonyms and performs stemming instead of just relying on exact word matching. ROUGE-L is the longest common sub-sequence based word matching. The longest co-occurrence in sequences of n-grams between generated text and reference texts are considered for calculating the score. To calculate these evaluation scores, we follow the script from Du et al. (2017), except for the corresponding scripts from other question generation models when ASGen is applied to them.

J Links to Downloadable Components

For Wikipedia data, we downloaded English Wikipedia dump in Feb 2019 from (https://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2). Page views were obtained from (https://dumps.wikimedia.org/other/pageviews/2019/2019-01/) and (https://dumps.wikimedia.org/other/pageviews/2019/2019-02/). For applying our method to other existing question generation models, we reproduce Zhao et al. (2018) using publicly available code (https://github.com/seanie12/neural-question-generation), Raffel et al. (2020) using publicly available code (https://github.com/patil-suraj/question_generation) and use the official code of Dong et al. (2019) (https://github.com/microsoft/unilm).