ASGEN: ANSWER-CONTAINING SENTENCE GENERATION TO PRE-TRAIN QUESTION GENERATOR FOR DATA AUGMENTATION IN QUESTION ANSWERING

Anonymous authors
Paper under double-blind review

ABSTRACT

Numerous machine reading comprehension (MRC) datasets often involve manual annotation, requiring enormous human effort, and hence the size of MRC data remains significantly smaller than that of those available for unsupervised learning, limiting the generalization capability. To overcome this issue, a new approach, which can generate synthetic question-and-answer data from large corpora such as Wikipedia, has been recently proposed. Such synthetic data can be utilized as additional data to pre-train the main MRC model before fine-tuning it using real, existing MRC data. However, the quality of generated question and answers are still far from being satisfactory, so that previous work introduced a pre-training technique for the question generator of this method by pre-training on the generation of next-sentence that follows the paragraph. However, the next sentence that follows a paragraph may have little relevance to the questions or answers from within the paragraph, and hence is not the ideal candidate for pre-training question generation. Thus, we propose a novel pre-training method called Answer-containing Sentence Generation (ASGen). Our approach is performed in multiple stages, involving two advanced techniques, (1) dynamically determining K answers from a given document and (2) pre-training the question generator using the task of generating the answer-containing sentence. We evaluate the question generation capability of our method by comparing the BLEU score with existing methods and test our method by fine-tuning the MRC model using the downstream MRC data after training on synthetic data. Experimental results show that our approach outperforms existing generation methods and increases the performance of the state-of-the-art MRC models across a range of MRC datasets such as SQuAD-v1.1, SQuAD-v2.0, KorQuAD, and QUASAR-T with no architectural modifications to the original MRC model.

1 INTRODUCTION

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

However, these datasets require a significant amount of human annotation to create pairs of a question and its relevant answer from a given context. Often the size of the annotated data is relatively small compared to that of data used in other unsupervised tasks such as language modeling. Hence, researchers often rely on the two-phase training method of transfer learning, i.e., pre-training the model using large corpora from another domain in the first phase, followed by fine-tuning it using the main MRC dataset in the second phase.

Most state-of-the-art models for MRC tasks involve such pre-training methods. Peters et al. (2018) present a bidirectional contextual word representation method called ELMo, which is pre-trained on a large corpus, and its learned contextual embedding layer has been widely adapted to many

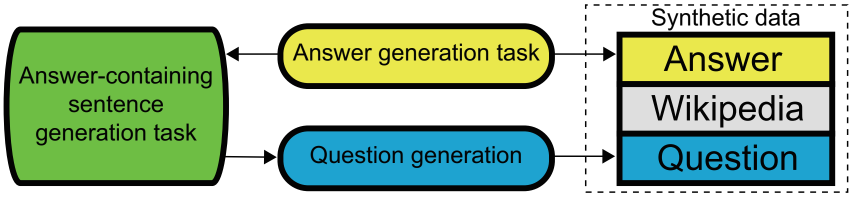


Figure 1: Overview of our Answer-containing Sentence Generation (ASGen) method.

other MRC models. Devlin et al. (2019a) show that pre-training with a masked language model on a large corpus and then fine-tuning on a downstream dataset results in significant performance improvements.

However, pre-training on another domain task and then fine-tuning on a downstream task may suffer from performance degradation, depending on which pre-training task is used in the first phase. For example, Yang et al. (2019) show that the pre-training task of next sentence classification decreases performance on the downstream MRC tasks. To handle this problem, generating synthetic data similar to the those of a downstream task is crucial to obtain a properly pre-trained model. Recently, researchers have studied a model for generating synthetic MRC data from large corpora such as Wikipedia. This is essentially a form of transfer learning, by training a generation model and using this model to create synthetic data for training the MRC model, before fine-tuning on the downstream MRC dataset.

Golub et al. (2017) suggest a two-stage synthesis network that decomposes the process of generating question-answer pairs into two steps, generating a fixed number (K) of answers conditioned on the paragraph, and question generation conditioned on the paragraph and the generated answer. Devlin et al. (2019b) introduced a pre-training technique for the question generator of this method by pre-training on the generation of next-sentence that follows the paragraph.

However, choosing a fixed number (K) of candidate answers from each paragraph will lead to missing candidates if K is too small, and will lead to having lower-quality candidates if K is too big. Moreover, the next sentence generation task is not conditioned on the answer, despite the answer being a strong conditional restriction for question generation task. Also, the next sentence that follows a paragraph may have little relevance to the questions or answers from within the paragraph, and hence is not the ideal candidate for pre-training question generation.

To address these issues, we propose Answer-containing Sentence Generation (ASGen), a novel pre-training method for a synthetic data generator with two novel processes, (1) dynamically predicting K answers to generate diverse questions and (2) pre-training the question generator on answer-containing sentence generation task. We evaluate the question generation capability of our method by comparing the BLEU score with existing methods and test our method by fine-tuning the MRC model on downstream MRC datasets after training on the generated data. Experimental results show that our approach outperforms existing generation methods, increasing the performance of the state-of-the-art MRC models across a wide range of MRC datasets such as SQuAD-v1.1, SQuAD-v2.0, KorQuAD, and QUASAR-T (Dhingra et al., 2017) without any architectural modifications to the MRC model.

2 Proposed Method

This section discusses the details of our proposed ASGen method. First, we will describe how the model generates synthetic data from Wikipedia. Next, we will explain the novel components of our method and how we pre-trained the question generator model based on them. Our model encodes paragraphs (contexts) in Wikipedia with two separate generation networks, the answer generator and the question generator.

Answer Generator. As shown in Fig. 2-(1), we generate the number of answer candidates K for a given context without the question by applying a fully connected feed-forward layer on the contextual embedding of classification token “[CLS]”. To make the contextual embeddings and to predict answer spans, we utilize a BERT (Devlin et al., 2019a) encoder (Fig. 2-BERT Encoder-A). Depend-

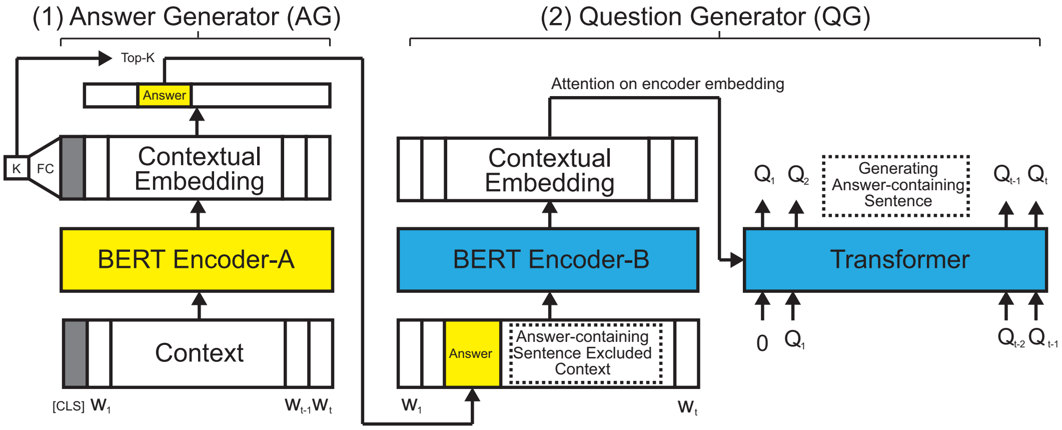

(1) Answer Generator (AG)
(2) Question Generator (QG)
Top-K
Answer
K
FC
Contextual Embedding
BERT Encoder-A
Context
[CLS] W1
Wt-1 Wt
Attention on encoder embedding
Contextual Embedding
BERT Encoder-B
Answer
Answer-containing Sentence Excluded Context
W1
Wt
Transformer
Q1 Q2
Generating Answer-containing Sentence
Qt-1 Qt
0 Q1
Qt-2 Qt-1

Figure 2: Architecture of our synthetic data generation method, (1) Answer Generator and (2) Question Generator. In the case of an answer-containing sentence generation task, the question generator takes the answer and the context without the answer-containing sentence as input and generates the answer-containing sentence.

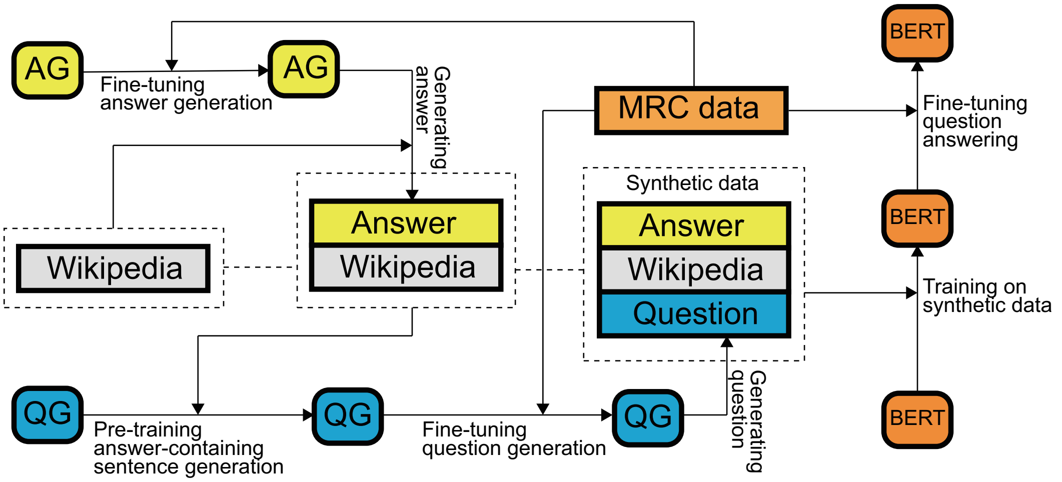

AG
Fine-tuning answer generation
AG
Generating answer
MRC data
BERT
Fine-tuning question answering
Wikipedia
Answer
Wikipedia
Synthetic data
Answer
Wikipedia
Question
BERT
Training on synthetic data
BERT
QG
Pre-training answer-containing sentence generation
QG
Fine-tuning question generation
QG
Generating question

Figure 3: Detailed procedure of Answer-containing Sentence Generation (ASGen) method, using the Answer Generator (AG) and the Question Generator(QG).

ing on the predicted number K, we select the K top candidate answer spans from the context. As shown in Fig. 2-(2), we use the K selected candidate answer spans as input to the question generator.

Question Generator. Next, as shown in Fig. 2-(2), we generate a question conditioned on the answer for each answer predicted from the answer generator. Specifically, we pass the context and an indicator for the answer span location in the context, to a BERT encoder (Fig. 2-BERT Encoder-Q). Then a Transformer decoder (Vaswani et al., 2017) generates the question word-by-word based on the encoded representation of the context and the answer-span. For pre-training this question generator on 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 questions and answers from a large corpus, such as paragraphs from all of Wikipedia in this paper. After generating the data, we train the MRC model on the generated data in the first phase, and then fine-tune on the downstream MRC dataset (such as SQuAD) in the second phase. In this paper, we use BERT as the default MRC model, since BERT represents the state-of-the-art model in many MRC datasets.

2.1 Dynamic Answer Prediction

The most natural method for humans to create a question-answer pair from a given context is to select the answer first, and then create a corresponding question. In this situation, we conjecture that a human is more likely to choose as an answer a phrase that is "answer-like", such as nouns, dates, names, etc., rather than choosing any random phrase. There may be several answers in the context that are likely to be selected by humans as answers, especially if the context is long or if it contains many nouns, dates, name, etc.

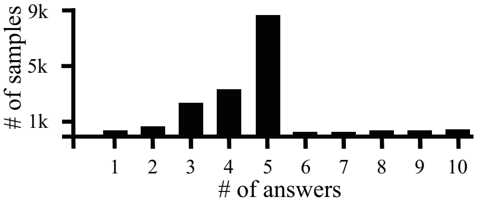


Figure 4: Distribution of the number of answers in each paragraph in SQuAD-v1.1 dataset.

For example, the context "Barack Hussein Obama II is an American attorney and politician who served as the 44th president of the United States from 2009 to 2017" can have as possible answers "Barack Hussein Obama", "44th", "United States", "2009 to 2017", etc. As shown in the Fig. 4, to see these characteristics, we examined the distribution of the number of answers in the SQuAD dataset, and hypothesize that there exists an underlying pattern in the number of answers that occur in a context. The conventional method to generate multiple answers from a context is to draw a fixed number (K) of answers. However, this approach can draw low-quality answers if K is too big, and it can unnecessarily reduce the size and diversity of the answers if K is too small.

Therefore, we predict the number of answers K in a given context $W = \{\mathbf{w}_t\}_0^T$ using regression as,

$$\{\mathbf{w}_t^{enc}\}_{t=0}^T = \text{BERT Encoder-A}(\mathbf{W})_t,$$
$$K = \lfloor f_k(\mathbf{w}_0^{enc}) \rfloor,$$

where T is the number of word tokens in the context with position 0 reserved for classification token '$[CLS]$', and f_k represents a fully connected unit with two hidden layers that have hidden dimension equal to H and 1 respectively, where H is the hidden dimension size of BERT Encoder-A.

To calculate the span start index score s_i for start index i, we use dot product of encoder output with a trainable start vector $\mathbf{S}$. For each start index i we calculate the span end index score $e_{i,j}$ for end index j in a similar manner with a trainable end vector $\mathbf{E}$, but conditional on i, i.e.,

$$\{s_i\}_{i=0}^T = \mathbf{S} \circ \mathbf{w}_i^{enc}$$
$$\{e_{i,j}\}_{i,j=0}^{T,T} = \mathbf{E} \circ f_s(\mathbf{w}_j^{enc} \oplus \mathbf{w}_i^{enc}),$$

where f_s represents a fully connected layer of hidden dimension H and $\oplus$ represents the concatenation operation. For training, we use the sum of mean square error loss on K and cross entropy loss on the s_i and $e_{i,j}$.

Next, we select the K top answer spans with the highest score summation of start index score and end index score, i.e.,

$$\begin{aligned} A^{span} &= \{(i,j) \mid 1 \leq i < T \text{ and } i \leq j < T\}, \\ a_k &= \max(\{a \mid \#\{(i,j) \mid (i,j) \in A^{span} \text{ and } s_i + e_{i,j} \geq a\}\}), \\ A_k^{span} &= \{(i,j) \mid (i,j) \in A^{span} \text{ and } s_i + e_{i,j} \geq a_k\}, \end{aligned}$$

The K selected answer spans A_k^{span} are passed to the question generator as input in the form of indication of the answer span location.

2.2 Pre-training Question Generator

In order to generate questions conditioned on various answers that may arise in a context, we generate a question for each of the K answers. In a previous study, Devlin et al. (2019b) proposed pre-training this generation model with an unsupervised task that generates the next sentence that follows a given paragraph to improve generation performance. We identify several issues with this approach. The final question generation task has the form of sentence generation given an answer and context, while next-sentence generation has no answer component. The next-sentence generation task is not conditioned on the answer, despite the answer being a strong conditional restriction for question generation task. Also, the next sentence that follows a paragraph may have little relevance to the questions or answers from within the paragraph, and hence is not the ideal candidate for pre-training question generation.

To solve these issues, we modify the context to exclude the sentence containing our previously generated answer, and pre-train our generator 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 leaving the answer to restructure the original context D to D^{ans} as,

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

Note that we change S^{ans} to not exclude the answer-containing sentence in the case of fine-tuning on the question generation, i.e.,

$$S^{ans} = \{(p,q,i,j)|p = i, q = j\}.$$

Then, we pass the previously generated answer to the sequence-to-sequence generation model as a segmentation encoding M^{ans} that identifies which part of the context is answer, i.e.,

$$M^{ans} = [\mathbf{m}_0 * p; \mathbf{m}_1 * (j - i); \mathbf{m}_0 * (T - q)], (p,q,i,j) \in S^{ans},$$

where $\mathbf{m}_0$ and $\mathbf{m}_1$ indicate trainable vector corresponding to segmentation id 0 and 1 respectively. Here we tag the segmentation id for each word in the context as 0 and each word in the answer as 1. A * B indicates the operation of concatenating vector A B times.

Next, we generate answer-containing sentence embedding $W^g = \{\mathbf{w}_t^g\}_0^T$ using a Transformer sequence-to-sequence model (encoder part is initialized with BERT) as,

$$\mathbf{w}_t^g = \text{Transformer Decoder}(\{\mathbf{w}_i^g\}_{i=0}^{t-1}, \text{BERT Encoder-Q}(D^{ans}, M^{ans})).$$

Finally, we calculate the loss of the generation model with cross-entropy over generated sentence words, i.e.,

$$\begin{aligned}
\{\mathbf{w}_t^o\}_{t=0}^T &= \{\text{Softmax}(\mathbf{w}_t^g E)\}_{t=0}^T, \\
\mathbb{L} &= -\left(\sum_{t=1}^{T}\sum_{i=1}^{D} \mathbf{y}_{t,i}\text{log}(\mathbf{w}_{t,i}^o) + (1 - \mathbf{y}_{t,i})\text{log}((1 - \mathbf{w}_{t,i}^o))\right)/T
\end{aligned}$$

where $\mathbf{y}$ indicates one-hot vector of ground-truth of answer-containing sentence word (in the case of fine-tuning, question word), D is vocab size and $E \in \mathbb{R}^{d \times D}$ represents word embedding matrix which is shared between BERT Encoder-Q and Transformer decoder.

In this way, we pre-train the model with a task that is similar to the final task of conditionally generating the question from a given answer and context.

3 EXPERIMENTAL SETUP

Wikipedia Preprocessing. To build the ASGen data and the synthetic MRC data, we collect all paragraphs from all articles of the entire English Wikipedia dump (Korean Wikipedia dump for KorQuAD) and generate questions and answers on these paragraphs. We apply extensive filtering and cleanup to only retain the highest-quality paragraphs from Wikipedia.

To filter out low-quality obscure pages, we remove all pages that received less than 200 cumulative page-views including all re-directions in a 2-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 all pages with less than 500 characters, as these pages are often low-quality stub articles, which removes a further 16% of the articles. We remove all "meta" namespace pages such as talk, disambiguation, user pages, portals, etc. as these often contain irrelevant text or casual conversations between editors.

In order to extract usable 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 other tables.

We then split the cleaned text from the pages into paragraphs, and remove all paragraphs with less than 150 characters or more than 3500 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, of the 60M paragraphs from the final 2.4M articles, our final Wikipedia dataset contains 8.3M paragraphs.

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. SQuAD-v1.1 (Rajpurkar et al., 2016) consists of crowd-sourced questions and answers based on contexts from Wikipedia articles. We also test the generation of unanswerable questions with SQuAD-v2.0 (Rajpurkar et al., 2018) dataset, which has extra unanswerable questions that are not present in SQuAD-v1.1. Similar to SQuAD-v1.1, KorQuAD (Lim et al., 2019) is another dataset created with the same procedure for Korean language. QUASAR-T (Dhingra et al., 2017) dataset contains human-annotated questions and answers but contexts are collected by an Information Retrieval (I.R.) system. To show that our generated data is robust and diverse enough to be effective on other unrelated datasets, we fine-tune and test the MRC model on QUASAR-T after training on the synthetic data generated using SQuAD-v1.1.

Implementation Detail. We use Tensorflow[1] to implement our model. For the answer generator, we use BERT (Devlin et al., 2019a) and two fully connected layers to predict the number of answers K. For the question generator, we use pre-trained uncased BERT (Base) as encoder and 12 layers of Transformer as decoder. For the generation of unanswerable questions as in SQuAD-v2.0, we separate unanswerable cases and answerable cases and train separate generation models. For the final MRC model, we use BERT (Large) which is the state-of-the-art model on multiple datasets with all official hyper-parameters.

Comparison with Other Models. We compare our question generation method, which is pre-trained on answer-containing sentence generation task (AS), with a method from Devlin et al. (2019b), which is pre-training on next-sentence generation task (NS), and with a method from Golub et al. (2017), which only trains question generation on final MRC dataset. We reproduced these methods as they were described in their original work for comparison. We generate synthetic data from Wikipedia using these generation models which are trained on the target downstream MRC datasets except for QUASAR-T. In the case of QUASAR-T, we use synthetic data which is generated by a model trained on SQuAD-v1.1. To check the effectiveness of our method on downstream MRC tasks, we evaluate our generated data on SQuAD-v1.1, SQuAD-v2.0, KorQuAD and QUASAR-T development sets by training state-of-the-art models (BERT and BERT+CLKT[2]) on generated data followed by fine-tuning on the train set for each dataset.

4 Quantitative Results

Dynamic Answer Prediction. We conducted an experiment to demonstrate the performance of our method in generating the number of answers in a given context. As shown in Table 1, in the case of fixed K, the mean distance from the ground-truth K^{gt} is the smallest at $K^{pred} = 5$ and the value is 0.91. In contrast, our answer generator generates a more appropriate number of answers

[1] `https://www.tensorflow.org`

[2] Internally developed model 'BERT+CLKT' which can be found on the leaderboard (`https://korquad.github.io/KorQuad%201.0`)

Table 1: Mean distance from prediction K^{pred} to ground-truth K^{gt}. The results are obtained on dev set of SQuAD-v1.1. N indicates the number of samples.

Model	$\sum(\lvert K^{pred} - K^{gt}\rvert)/N$
Fixed-K ($K^{pred} = 5$)	0.91
Dynamic-K Answer Prediction	0.72

Table 2: Comparison of BLEU-4 scores of sentences (pre-train) and questions (fine-tune) generated by different pre-training methods, i.e, without pre-training, pre-training on NS and pre-training on AS.

Model	BLEU-4 score	
	Pre-train on Wikipedia	Fine-tune on SQuAD
Only trained on SQuAD (Golub et al., 2017)	N/A	18.1
Pre-train on NS (Devlin et al., 2019b)	3.4	19.7
Pre-train on AS	11.3	25.9

than the fixed K approach, while reducing the mean distance between the ground-truth K^{gt} and the prediction K^{pred} as 0.72.

Pre-training Question Generation. We compare the BLEU-4 score of the generated sentences after pre-training and the BLEU-4 score of the generated questions after fine-tuning on SQuAD to see the performance of our question generation. As shown in Table 2, our method (AS) scored 7.9 points higher than NS in sentence generation. This can be interpreted as less uncertainty in text generation because in the case of AS, we provide a clue (answer) for text generation to the model. Finally, AS outperforms the previous method NS by 6.2 points in BLEU-4 score on generation of SQuAD-v1.1 questions.

Downstream Task Performance. We conduct experiments by training MRC models on the synthetic data generated by our method (AS) from Wikipedia before fine-tuning the model on the downstream dataset to show the effectiveness of our synthetic data generation. For each dataset, the question generator model is pre-trained with AS and fine-tuned on the downstream data. As shown in Table 3, our method shows an improvement of 1.9 F1 score on SQuAD-v1.1, 4.0 F1 score on SQuAD-v2.0 and 0.5 F1 score on KorQuAD from the state-of-the-art baseline models. Our method also outperforms NS on both SQuAD-v1.1 and SQuAD-v2.0.

Effect of MRC and Synthetic Data Size. We conducted experiments with varying amount of downstream MRC data. In each case, the generator model and the MRC model were both trained on the corresponding limited data. As shown in Fig. 5-(a), our method (AS) consistently outperforms NS. This performance difference is particularly significant for smaller data sizes. We also conducted experiments by training the MRC model with varying amounts of generated data for the same number of steps. As shown in Fig. 5-(b), larger size of generated data consistently gives better performance.

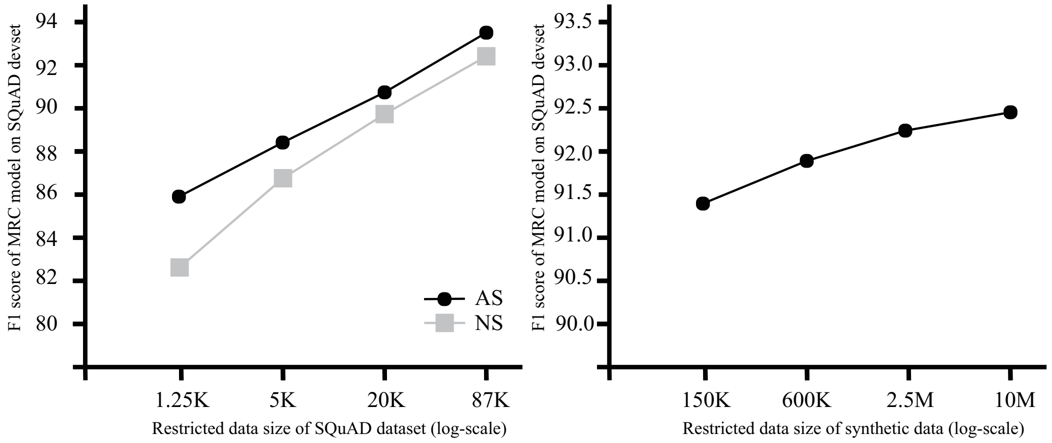
Transfer Learning to Other Dataset. We trained BERT on synthetic data from the generation model fine-tuned on SQuAD-v1.1, followed by fine-tuning on QUASAR-T to see if the generated data is robust and diverse enough to be directly useful on other datasets. QUASAR-T has two separate datasets, one with short snippets as context, and other with long paragraphs as context. As shown in Table 4, training with our synthetic data shows an improvement of 2.2 F1 on the test set for the short case, and 1.7 F1 on test set for the long case.

5 QUALITATIVE RESULTS

Comparison of Question Generation. We qualitatively compare the generated questions after pre-training with NS and AS to demonstrate the effectiveness of our method. For the correct answer "49.6%" as shown in the first sample in Table 5, NS omitted "Fresno", which is a necessary condition to make the question specific, while AS'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,

Table 3: Comparison of EM/F1 scores of fine-tuned BERT model on SQuAD v1.1, SQuAD v2.0 and KorQuAD development datasets using corresponding synthetic data for training.

Model + pre-training method	SQuAD v1.1		SQuAD v2.0		KorQuAD	
	EM	F1	EM	F1	EM	F1
BERT (Baseline) (Devlin et al., 2019a)	83.9	90.9	78.8	81.8	-	-
BERT + SynNet (Golub et al., 2017)	85.1	91.4	80.9	83.9	-	-
BERT + NS (Devlin et al., 2019b)	85.6	92.3	81.5	85.1	-	-
BERT + AS	86.3	92.8	82.5	85.8	-	-
BERT + CLKT (Baseline)	-	-	-	-	87.1	94.5
BERT + CLKT + AS	-	-	-	-	87.8	95.0



(a) Varying amount of MRC data (SQuAD-v1.1).

(b) Varying amount of synthetic data.

Figure 5: Comparison of F1 scores of MRC model on SQuAD-v1.1 development set with different data sizes. Each experiment is restricted to use only the corresponding amount of data for generation and training.

where NS uses the word “available” rather than the more relevant words from the answer-containing sentence, but AS uses many of these words such as “most” and “popular” to make more contextual questions. Also, the question from NS asks about “two” libraries, while the answer has “three” libraries, showing the lack of sufficient conditioning on the answer. The third sample also shows that AS draws more context-related questions by drawing the exact subject “TARDIS” to use for the corresponding answer in a similar vein.

6 Related Work

Question Answering. Question answering is one of the most active research areas in NLP. For MRC tasks, a large number of datasets were proposed, most often focused on finding an answer span for a question from a given paragraph. Among these, the most popular and fully human-annotated datasets are SQuAD-v1.1 (Rajpurkar et al., 2016), SQuAD-v2.0 (Rajpurkar et al., 2018), KorQuAD (Lim et al., 2019) and HotpotQA (Yang et al., 2018). However, these datasets are small with around 100K samples each, which is far smaller than the datasets used for unsupervised tasks such as language modeling. Researchers have also proposed some large datasets such as QUASAR-T (Dhingra et al., 2017) and TriviaQA (Joshi et al., 2017). Although these have human-made questions and answers, the relevant paragraphs are retrieved using IR systems, often resulting in unrelated paragraphs or paragraphs which lack sufficient information to answer the question correctly. In order to overcome these drawbacks, we focus on the task of generating data which is comparable to human-annotated data in quality while also enabling generation of large quantities of MRC data.

Table 4: Comparison of EM/F1 scores of fine-tuned BERT model on QUASAR-T dataset. The synthetic data is generated from the generation model trained on SQuAD-v1.1.

Model + pre-training method	Short(Dev)		Short(Test)		Long(Dev)		Long(Test)	
	EM	F1	EM	F1	EM	F1	EM	F1
BERT (Devlin et al., 2019a)	74.3	78.6	74.1	77.8	72.1	75.6	72.1	74.8
BERT + AS	76.5	80.1	76.5	80.0	74.2	77.4	73.8	76.5

Table 5: Examples from SQuAD-v1.1 development dataset demonstrating generated questions. We compare our method (AS) with NS. Colored Text indicates the answer given for question generation.

Context	The 2010 United States Census reported that Fresno had a population of 494,665. 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%) African American, 8525 (1.7%) Native American ... (omit)
Question by NS	What percent of the population is White?
Question by AS	What percentage of the Fresno population is White?
Context	(omit) ... According to the American Library Association, this makes it the largest academic library in the United States, and one of the largest in the world. Cabot Science Library, Lamont Library, and Widener Library are three of the most popular libraries for undergraduates to use, with ... (omit)
Question by NS	Which two libraries are available for undergraduates to use?
Question by AS	What are the three most popular libraries for undergraduates?
Context	(omit) ... He fled from Gallifrey in a stolen Mark I Type TARDIS "Time and Relative Dimension in Space" time machine which allows him to travel across time and space. The TARDIS has a "chameleon circuit"...(omit)
Question by NS	What does the doctor refer to?
Question by AS	What does the TARDIS stand for?

Transfer Learning. Pre-training methods are extensively used in NLP to obtain better word representations. The usual approach is to fine-tune a layer that is pre-trained on another task such as language modeling. ELMo (Peters et al., 2018) learn a language model with bi-directional LSTM (Hochreiter & Schmidhuber, 1997) and then fine-tunes its representation on the downstream task by concatenating the representations of its hidden layers to the input representation of other models. Open-GPT (Radford et al., 2018) uses a Transformer decoder to learn a language model on a large dataset followed by fine-tuning on the downstream task. BERT (Devlin et al., 2019a) and MASS (Song et al., 2019) learn forms of masked language modeling using Transformer blocks and then fine-tune on the downstream task. Similarly, XLNet (Yang et al., 2019) uses the Transformer decoder but learns permutation language modeling. While our approach is similar to these, our pre-training task for question generator generates answer-containing sentences to make pre-training similar to the final question generation task.

Synthetic Data Generation. Golub et al. (2017) proposed separating the answer generation and the question generation. Their proposed model generates questions conditioned on generated answers, and then they evaluate the quality of the synthetic data by training an MRC model with it before fine-tuning on SQuAD. We improve the performance of their approach by designing a new pre-training technique for the question generation task to produce better quality of synthetic data than existing methods.

7 Conclusion

We propose two advanced training methods for generating high-quality and diverse synthetic data for MRC. Firstly, dynamically choosing K top answer spans from an answer generator, and then generating the sentence containing the corresponding answer span as a pre-training task for the question generator. Using the proposed methods, we generate 43M synthetic training samples and train the MRC model before fine-tuning on the downstream MRC dataset. Our proposed method improves the performance of the state-of-the-art models on SQuAD-v1.1, SQuAD-v2.0, KorQuAD, QUASAR-T datasets without any architectural modification to the MRC model.

REFERENCES

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2019a.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *URL https://nlp.stanford.edu/seminar/details/jdevlin.pdf*, 2019b.

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

David Golub, Po-Sen Huang, Xiaodong He, and Li Deng. Two-stage synthesis networks for transfer learning in machine comprehension. In *Proceedings of the conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 835–844, 2017.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. In *Neural Computation*, volume 9, pp. 1735–1780. MIT Press, 1997.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1601–1611, 2017.

Seungyoung Lim, Myungji Kim, and Jooyoul Lee. Korquad1.0: Korean qa dataset for machine reading comprehension. *arXiv preprint arXiv:1909.07005*, 2019.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pp. 2227–2237, 2018.

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

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2383–2392, 2016.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 784–789, 2018.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mass: Masked sequence to sequence pre-training for language generation. In *International Conference on Machine Learning (ICML)*, pp. 5926–5936, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 5998–6008, 2017.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2369–2380, 2018.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*, 2019.