Regularization of Distinct Strategies for Unsupervised Question Generation

Junmo Kang* **Giwon Hong*** **Haritz Puerto San Roman*** **Sung-Hyon Myaeng**
School of Computing, KAIST
Daejeon, Republic of Korea
{junmo.kang, gch02518, haritzpuerto, myaeng}@kaist.ac.kr

Abstract

Unsupervised question answering (UQA) has been proposed to avoid the high cost of creating high-quality datasets for QA. One approach to UQA is to train a QA model with questions generated automatically. However, the generated questions are either too similar to a word sequence in the context or too drifted from the semantics of the context, thereby making it difficult to train a robust QA model. We propose a novel regularization method based on teacher-student architecture to avoid bias toward a particular question generation strategy and modulate the process of generating individual words when a question is generated. Our experiments demonstrate that we have achieved the goal of generating higher-quality questions for UQA across diverse QA datasets and tasks. We also show that this method can be useful for creating a QA model with few-shot learning.

1 Introduction

Machine Reading for Question Answering (MRQA) is the task of answering questions from a context that contains the answer. This field has seen remarkable progress in recent years, with QA models outperforming humans on a question answering (QA) benchmark like SQuAD (Rajpurkar et al., 2016).

Training a QA model requires a large amount of data, and constructing such a dataset is usually laborious or sometimes even impossible for some domains and languages. Because of this, Lewis et al. (2019) explored unsupervised question answering (UQA), a setting where manually constructed triples, *(context, question, answer)*, are not available for training. They approached the problem with unsupervised question generation (UQG), where answers are first extracted from contexts, and then questions are generated from *(contexts, answers)* pairs. Such questions are combined into *(contexts, questions, answers)* triples to form a training dataset for a QA model.

Context	
... Level 1 of DDM Architecture was formally published in 1986. ...	
Generated Question	
LM-type	**Copy-type**
When did the first **level 1 of DDM Architecture** *come out?*	*When* **level 1 of DDM Architecture was formally published***?*

Figure 1: An example of **LM-type** (**left**) and **copy-type** (**right**) questions generated from a context. Newly generated tokens are in *italics* and copied tokens in **bold**.

Another approach to UQG is based on language models (LM). Radford et al. (2019) proposed an LM approach, GPT-2, that shook the natural language processing (NLP) research community with its remarkable capability of automatically generating high-quality text. Naturally, GPT-2 was shown to generate high-quality questions from contexts (Klein and Nabi, 2019; Puri et al., 2020).

Despite the recent progress made by these efforts, UQG remains to be an open problem. For example, the method proposed by Lewis et al. (2019) generates a question that is often so similar to a word sequence in the given context that QA models are trained with only straightforward questions and thus hampered from solving more challenging problems. We refer these questions to *copy-type*. The drawback of copy-type questions can be mitigated by generative LM like GPT-2 as they can generate tokens that are not in the context but semantically plausible. However, they can generate questions semantically drifted from the context which may not be answerable even by humans. Figure 1 shows

*Equal contribution.

Findings of the Association for Computational Linguistics: EMNLP 2020, pages 3266–3277

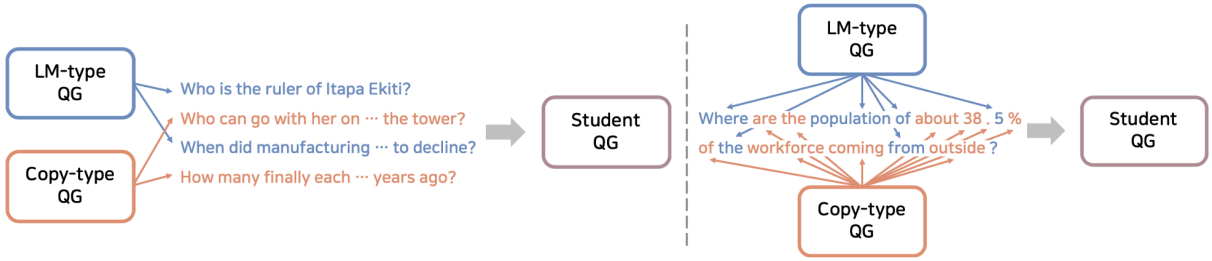

Figure 2: Comparison between instance-level and token-level regularization. The entire questions are selected for question generation at the instance-level whereas individual tokens are selected at the token-level.

an example for LM-type and copy-type questions.

Inspired by the knowledge distillation approach with teacher-student interactions (Hinton et al., 2015), we propose a pipeline architecture where a student module learns from a teacher how to generate higher quality questions that mitigate the drawbacks of both strategies of questions. The teacher employs two QG models[1] (assistants), LM-type and copy-type, and adopts a *semantic-level regularization* process newly designed to avoid a bias toward a particular question generation strategy (i.e., either copy-type or LM-type). The student module learns how to make a balance between the two extreme types of questions generated by the assistants. The regularization helps suppress the copying behavior and semantic drifts of the existing QG models and generate more versatile questions that are compatible with the given contexts.

Our contributions are i) a novel generation method based on the teacher-student architecture that regularizes two generation models in an unsupervised setting. ii) adopting this method for unsupervised question generation to create a higher-quality QA dataset[2] compared to existing UQG models without manual labeling. iii) using this QA dataset to train a QA model and demonstrate the robustness and effectiveness of our generation method thorough experiments in low QA data regimes.

2 Preliminary Study

In order to ensure that using the two somewhat conflicting types of questions can result in a QA improvement, we set out to run a preliminary experiment. We simply combine the two datasets, copy-type and LM-type (question creation details are provided in Section 4.1) to train a QG model to see if the regularization idea would be beneficial at all. We posit that a positive result would not only justify a more sophisticated regularization process but also serve as a baseline for the experiments.

Since the dataset used for training the QG model contains two types of questions, we use the following batch loss:

$$\mathcal{L}_{batch} = \alpha \mathcal{L}_{LM} + (1 - \alpha)\mathcal{L}_{copy} \quad (1)$$

where the hyperparameter α controls the loss incurred by each question type in the batch. Assuming that learning the patterns of copy-type questions would be easier than LM-type questions, due to the simplicity of the former, we enforce that reducing the total loss is influenced more by the LM-type rather than the copy-type.

Model	EM	F1
Copy-type QG	40.1	49.4
LM-type QG	42.0	50.9
QG trained on copy and LM	44.4	53.9

Table 1: A result of the preliminary experiment with QA on the SQuAD 1.1 dev set.

To test the quality of this naive QG approach, we have trained three QA models based on three different QG models: i) a QG model trained on 10K copy-type questions, ii) a QG model trained on 10K LM-type questions, iii) a QG model trained on the combination of 5K copy-type questions and 5K LM-type questions. To ensure a fair comparison, the three models use the same contexts and answers. As shown in Table 1, the combination of the two types of questions yields better performance. This

[1]The proposed model can be easily extended to more than two strategies.

[2]https://github.com/HaritzPuerto/UQA

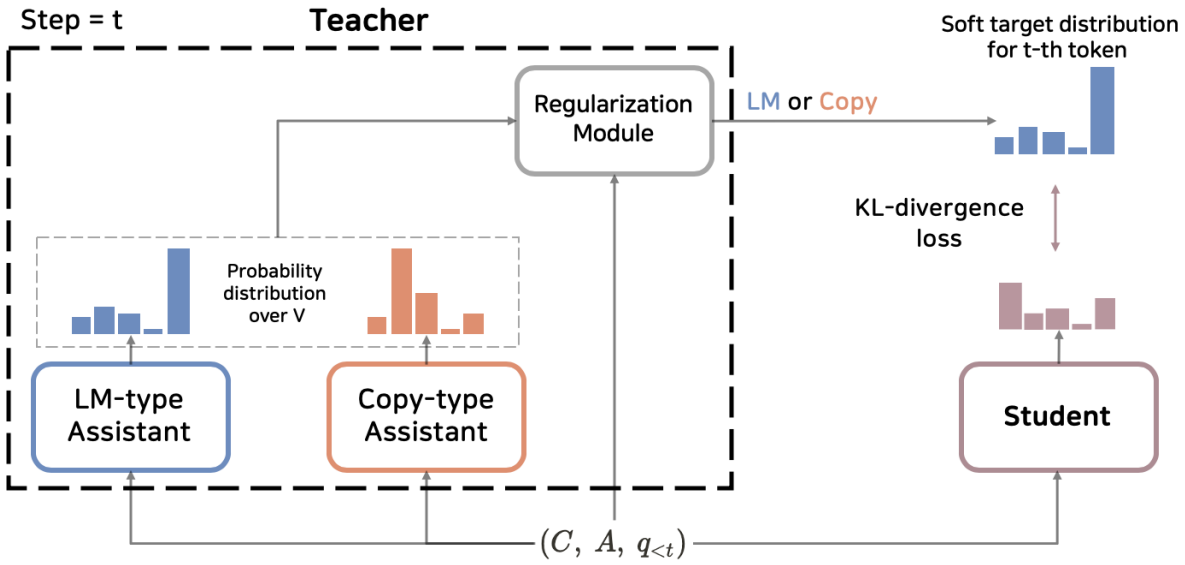

Figure 3: Overall structure of the proposed approach. At time step t, the pre-trained LM-type and copy-type assistants produce each probability distribution over vocabulary for the token q_t. Then, the regularization module selects the probability distribution $P(w)$ for the token q_t such that the generated question is not biased to either type, and $P(w)$ is used as the soft target to train the student module.

positive result sets the stage for the main thrust of this work: design and application of *token-level regularization* for training a QG model to overcome the drawbacks of two styles of questions.

3 Proposed Approach

In an unsupervised setting, it is difficult to solve the problems related to the LM-type and copy-type QG models, primarily because the gold standard questions are not provided. However, the preliminary study (Section 2) shows there is evidence that using two QG models, LM-type and copy-type, in an interleaving way has the potential to solve the problems of each model. Instead of simply merging two datasets (as in the Preliminary Study), we propose a finer-grained approach of using token-level regularization rather than instance-level that selects full questions. Figure 2 illustrates the difference.

We posit that generated questions should not be easily classified into either LM-type or copy-type if the two styles were to be inter-mixed in a balanced way. In other words, the style of the questions should not be highly biased toward either side. In short, our unsupervised QG problem is reduced to devise a token-level regularization method that generates questions indistinguishable between the two question types and eventually mitigates their drawbacks.

3.1 Problem Formulation

The goal of the QG model is to generate the most probable question $Q = (q_1, ..., q_{|Q|})$ given a context $C = (c_1, ..., c_{|C|})$ and an answer $A = (a_1, ..., a_{|A|})$, which is a subspan of C, (i.e., $a_1 = c_i$ and $a_{|A|} = c_j$ where $1 \leq i \leq j \leq |C|$).

$$\hat{Q} = \arg\max_{Q} P(Q|C, A) \tag{2}$$

As discussed in Section 2, $\hat{Q}$ should be discouraged from being biased toward either LM-type or copy-type. To enforce this property, we define a function $\mathcal{F}$ that returns the probability of the question following a specific $type$ (LM for LM-type or CP for copy-type), given the context and the generated question tokens up to t.

$$\mathcal{F} : Q \times C \times type \rightarrow [0, 1] \tag{3}$$

where Q and C represent the infinite set of possible questions and contexts, respectively.

Question tokens are generated sequentially by considering the sub-sequence of up to $t-1$ tokens at time step t (denoted as $q_{<t}$) that have been generated so far. With the binary function defined above, we can constrain the next token $\hat{q}_t$ to satisfy the following condition: if the question generated up to $t-1$ is of LM-type, $\hat{q}_t$ should maximize the

score of the copy-type, or vice versa.

$$\hat{q}_t = \begin{cases} \arg\max_w \mathcal{F}(q_{<t} : w, LM), & \text{if } \mathcal{F}(q_{<t}, LM) \\ & < \mathcal{F}(q_{<t}, CP) \\ \arg\max_w \mathcal{F}(q_{<t} : w, CP), & \text{otherwise} \end{cases} \quad (4)$$

where w is a token over the vocabulary V and ":" represents the concatenation operation.

3.2 Overall Architecture

As a way to implement the aforementioned idea of generating each token $\hat{q}_t$ sequentially based on the nature of the sub-question up to $t-1$, we take a teacher-student structure as a pipeline inspired by Hinton et al. (2015). The teacher is composed of two types of "assistants" (LM-type QG and copy-type QG models), and a regularization module, whereas the student is a single QG model that receives the knowledge transferred from the teacher. In this way, the student model can learn the regularization process of the teacher and generalize it even without golden labels.

The pipeline system works as follows. Given a context as the only input, it first randomly selects a named-entity as the answer and predicts *wh*-word to input to the two assistants (i.e., the two QG models). This *wh*-word prediction is needed for a performance boost as shown in (Kang et al., 2019). Each of the two assistants predicts a new token, and the regularization module subsequently selects the one to accept. The selected token is input again to the two assistants with the previously generated tokens to generate the next ones and so on until a question mark is generated. At each time step t, we also store the probability distribution over all the tokens in the vocabulary, from the selected assistant. These distributions, rather than the tokens, help generalize the knowledge of the teacher and hence allow the student to learn the probabilities of selecting particular tokens at individual time steps in a more reliable way. This entire procedure is illustrated in Figure 3, as well as in Algorithm 1 in Appendix A.3.

3.3 Question Generation Module

One of the benefits of our architecture is that the modules are not bounded by any specific model. For the current work, we employ the QG model proposed by Chan and Fan (2019) for the two assistants of the teacher and for the student, taking advantage of BERT (Devlin et al., 2019).

The essence of this QG model is to input a context, an answer, generated question tokens at each time step, and a `[MASK]` token at the end. The embedding of the last token (i.e., [MASK]) at the output layer is used to predict the next generated token, $\hat{q}_t \in V$, where V is the vocabulary.

The QG model works as follows. First, the context C and the answer A, which is a sub-span of the context, are integrated into C' with the special tokens `[HL]` to signal the start and end of the answer.

$$C' = \Big[c_1, c_2, ..., [HL], a_1, ..., a_{|A|}, [HL], ..., c_{|C|}\Big] \quad (5)$$

Then the question tokens generated so far (prior to the current time step t) are added to complete the input to BERT, which is used to generate H:

$$X_{t-1} = ([CLS], C', [SEP], \hat{q}_1, ..., \hat{q}_{t-1}, [MASK]) \quad (6)$$

$$H = BERT(X_{t-1}) \quad (7)$$

where $H \in \mathbb{R}^{|X_t| \times h}$ is the matrix of BERT token embeddings and h is the hidden size of a BERT token embedding.

In order to generate a token from the BERT output, the embedding of the `[MASK]` token, $H_{[MASK]}$, is transformed into the vocabulary space using the linear layer $W \in \mathbb{R}^{h \times |V|}$. This gives a probability distribution over the vocabulary given the input:

$$P(w|X_{t-1}) = softmax(H_{[MASK]} \cdot W + b) \quad (8)$$

The next token, $\hat{q}_t$, is the word, $w \in V$, with the highest probability:

$$\hat{q}_t = \arg\max_w P(w|X_{t-1}) \quad (9)$$

3.4 Teacher Module

The teacher module consisting of two assistant modules and the regularization module provides soft targets to the student module. The assistant modules are pre-trained with cross-entropy to implement the question generators and serve as the source of knowledge to the regularization module.

Inspired by GAN (Goodfellow et al., 2014), we set the goal of the regularization module to preventing the generated question from being easily detected as either LM type or copy type. By making a generated question indistinguishable between the LM and copy types, we attempt to make the student mitigate the drawbacks of either type.

We implement the regularization module as a discriminator, $\mathcal{D}$, that takes as input the context and

the list of generated tokens up to $t-1$, and decides which of the LM-type, θ_{LM}, and copy-type, θ_{CP}, assistants should generate the next token. More formally, the next token is selected as follows:

$$\mathcal{D} : C \times Q \rightarrow \{LM, CP\} \quad (10)$$

$$\hat{q}_t = \begin{cases} \arg\max_w P(w|X_{t-1}, \theta_{LM}), & \text{if } \mathcal{D}(C, q_{<t}) = LM \\ \arg\max_w P(w|X_{t-1}, \theta_{CP}), & \text{if } \mathcal{D}(C, q_{<t}) = CP \end{cases} \quad (11)$$

This discriminator is implemented using BERT. Its input, X_{t-1}, is the concatenation of the question tokens with the context:

$$X_{t-1} = ([CLS], \hat{q}_1, ..., \hat{q}_{t-1}, [SEP], C, [SEP]) \quad (12)$$

The BERT embedding of the token `[CLS]` is then input to the binary classifier $W_D \in \mathbb{R}^{h \times 2}$ to select the type.

$$H = BERT(X_{t-1}) \quad (13)$$

$$type = \arg\max(\sigma(H_{[CLS]} \cdot W_D + b_D))) \quad (14)$$

where σ is the sigmoid function.

The training dataset for the regularization module consists of a list of (context, question from either LM-type or copy-type assistant) pairs serving as input and class, either LM-type or copy-type, as a label. Each question is truncated to a random length to simulate the use case of sequentially generating question tokens.

This module achieves a regularization effect because it predicts the class of a question: if the class of the generated question up to some time step t is predicted to be an LM-type, the next token of the question is generated from the copy-type assistant. This regularization process will be reflected in the questions used to train the student, which in turn learns how to generate questions that are regularized between the LM style and the copy style.

3.5 Student Module

Following Hinton et al. (2015), the student is trained using the probability distributions of generating tokens by either one of the assistants as soft targets. This allows the student to learn more information from the teacher than using hard targets (i.e., tokens) because it can learn even a small probability of generating a token. Since we regularize the probability distribution to be given to the student module at each time step, the student ends up learning how to generalize the regularization process.

We use KL-divergence loss to minimize the difference between the teacher's (i.e. from one of the assistants) and student's probability distributions as follows.

$$\mathcal{L}_{student} = D_{KL}(P(\hat{w}|X_{t-1}; \theta_{student}) \,||\, P(w|X_{t-1}; \theta_{teacher})) \quad (15)$$

It is worth noting that the training with soft targets is possible because the same architecture and thus, the same vocabulary is used for the assistants and the student.

To avoid repeated tokens, the penalized sampling technique as in Keskar et al. (2019) is applied for the calculation of the probability distribution. This technique is also applied to prevent from generating special tokens and the answer, which should not be included in a question. Details of the penalized tokens are provided in Appendix A.2.

4 Experiments

The primary goal of our experiments is to demonstrate that the method of regularizing copy-type and LM-type questions yields questions that overcome the drawbacks of the two types. By measuring the QA performance on several QA datasets, we establish the generalizability of the proposed approach.

4.1 Experimental Setting

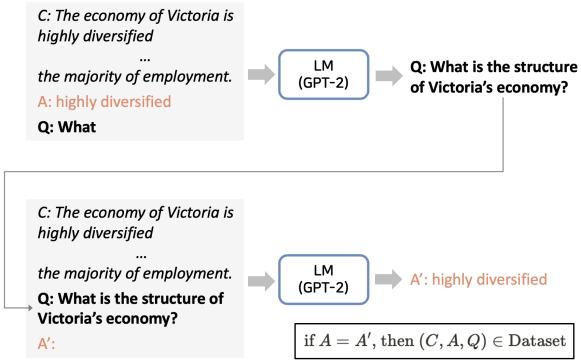

Figure 4: LM-based QG Filtering.

Dataset Creation All our models use contexts from the dataset of Lewis et al. (2019)[3]. In the case of the copy-type dataset, it also uses the questions and answers from the original dataset to train a QG model to create the copy-type questions. On the other hand, the QG-type dataset uses Stanza (Qi et al., 2020) to extract named-entities to obtain

[3]https://github.com/facebookresearch/UnsupervisedQA

Model	SQuAD	NewsQA	TriviaQA	SearchQA	HotpotQA	Natural Questions
Lewis et al. (2019)	54.3	27.4	31.5	**34.6**	31.8	25.2
LM-type QG	50.9	28.2	31.0	17.0	31.7	36.9
Copy-type QG	49.4	22.6	26.5	31.2	25.6	20.7
Teacher	58.2	29.8	33.1	12.6	**34.5**	36.7
Student	**60.2**	**30.4**	**36.8**	18.3	33.2	**37.3**

Table 2: F1 scores of the baselines including SOTA on SQuAD and the two QG models, and our proposed approach (Teacher and Student) on the in-domain MRQA shared task datasets. The Student is our final QG model.

answers and GPT-2 to create the questions. Figure 4 illustrates the process of LM-type question generation. Since GPT-2 is not optimized to generate questions (Klein and Nabi, 2019), we add a question filter to eliminate potentially unanswerable questions as a way to maximize its QA performance. The filter accepts a (C, A, Q) iff the question generated by GPT-2 is answerable by itself. A further explanation is in Appendix A.1. All the generated datasets, i.e. intermediate datasets to reproduce LM-type, Copy-type, teacher and student QGs, and the dataset to train the final QA model, are publicly available in our GitHub repository.

Implementation The hyperparameter α used in Eq. (1) is set to 0.8. The QG models are based on BERT-HLSQG (Chan and Fan, 2019), with the hyperparameters provided by the authors and minor modifications to apply penalized sampling as in Keskar et al. (2019).

The discriminator model and the QA model for the evaluation are based on the BERT-base and BERT-large (Devlin et al., 2019), respectively, implemented by Hugging Face (Wolf et al., 2019). Their default hyperparameters are used without dropout. A detailed description of the hyperparameters is in Appendix A.4.

Training The copy-type and LM-type assistants are pre-trained on 10K instances. The discriminator model is trained on a fully balanced training set of 70K instances. The student QG model is trained on 10K instances generated by the teacher module. The QA models for the experiments are trained with 10K instances.[4] The contexts for the training datasets are randomly sampled without replacement.

Evaluation The in-domain MRQA shared task (Fisch et al., 2019)[5] including modified SQuAD 1.1 (Rajpurkar et al., 2016), NewsQA (Trischler et al., 2017), TriviaQA (Joshi et al., 2017), SearchQA (Dunn et al., 2017), HotpotQA (Yang et al., 2018), and Natural Questions (Kwiatkowski et al., 2019) is used for the experiments in Sections 4.2, 4.3, and 4.5. The SQuAD 1.1 dev set[6] is used for the experimental results in Section 4.4.

4.2 Overall Performance

For a fair and extensive comparison with the previous work from Lewis et al. (2019) on MRQA datasets, a QA model was trained using the dataset provided by Lewis et al. (2019). In addition, to understand the effect of each module in the entire system, we trained the QA models with the datasets generated by the two assistants, the teacher module, and the student module.

As can be seen in Table 2, our model significantly outperforms the baselines on all the QA datasets but SearchQA. This result validates the proposed approach, suggesting it alleviates the drawback of the copy-type questions generated by (Lewis et al., 2019) as well as those generated by the single QG models. The proposed approach clearly creates a more challenging and yet semantically less deviating QA dataset that ends up training a more robust and reliable QA model. The Exact Match (EM) scores are provided in the appendix A.5.

The reason for the low performance of our approach on SearchQA appears to lie in the nature of this dataset. Considering that the LM-type QG is significantly interior to the copy-type QG on the dataset only, it is clear that copy-type questions are much more useful for SearchQA than the LM-type. Given that our models ("Teacher" and "Student")

[4]The training of the discriminator, QG, and QA models takes 4, 9, and 0.6 hours on a Tesla P100, respectively.

[5]https://github.com/mrqa/MRQA-Shared-Task-2019

[6]https://rajpurkar.github.io/SQuAD-explorer/

Module	EM	F1
Random	46.6	56.5
Frequency-based	47.7	57.8
Discriminator	**49.8**	**60.2**

Table 3: Comparison among different regularization schemes that affect the student QG.

attempt to deviate from the copy-type questions by design, the generated questions end up making the QA model less effective in handling copy-type questions. The student model performs better than the teacher module due to the student's capability to generalize the discriminator, which is based on our heuristic assumption (Eq. 11).

4.3 Roles of the Modules

Since the teacher module can generate questions by itself, we show its performance in Table 2. Across most of the datasets, it outperforms the baselines including the two QG models in the assistants, showing the clear value of the regularization module that effectively combines them. By comparing its performance on SQuAD against the result in the preliminary study (Table 1), we confirm that the token-level regularization (58.2 in F1) is more effective than the instance-level regularization (53.9) implemented by the "QG trained on copy and LM" model, as suggested in Section 3.

Table 2 also witnesses the value of the student module or the teacher-student architecture as it helps improve the performance over the teacher module; the student learned and generalized the regularization process from the teacher.

4.4 Effect of Regularization Module

The discriminator model was trained on 35K LM-type questions and 35K Copy-type questions (70K in total, as mentioned in section 4.1). After the training, the discriminator model achieves an accuracy of 95.53% on a 10K dev set (5K for each type). To further study the effect of the discriminator model in regularizing the two styles of questions, we replace it with simpler models and analyze the performance of the generated questions. The simpler models are i) random, i.e., at each time step, an assistant is selected randomly, and ii) frequency-based, i.e., the probability of selecting an assistant is proportional to the number of times the other assistant has been selected. As a result, three different sets of questions are used for training a QA model and evaluated on the dev set of SQuAD 1.1.

As shown in Table 3, the discriminator model outperforms the other two models. This indicates that the context information and the previously generated question tokens are essential to correctly determining the question type, which in turn affects the quality of the regularization. This result is coherent with the intuition that the context must be essential to deciding whether or not a question is a copy from itself.

Also noteworthy is that the frequency-based model performs better than the random model, implying that frequency plays a role in the regularization task. It appears that striking a balance between the two styles in the generating tokens at each step helps in generating higher quality questions. The random model is likely to generate a less balanced list of tokens, e.g., the first half of the tokens biased towards one type. The result reaffirms that the generated question should not be heavily biased toward a question type.

4.5 Potential for Few-shot Learning

Although our main goal is to improve unsupervised QA through a dataset generated in an unsupervised way, it is instructive to consider a few-shot learning setting, in which a limited number of pre-labeled training instances are used instead of the entire set. To investigate the potential of the proposed method for improved QA performance with few-shot learning, we pre-train the BERT-large QA model with a synthetic QA dataset generated from the student module. The QA model is then fine-tuned with increasing numbers of pre-labeled instances, starting from 0 (zero-shot learning) to all the available training instances. Figure 5 shows the result on the in-domain MRQA shared task.

As can be seen, UQG is remarkably useful in the no-data or small-data regimes. The proposed QG method added to the BERT-large QA model enjoys a performance boost of about 10 to 20 points, depending on the datasets, with an addition of only about 100 pre-labeled instances. On the other hand, it is sufficient to add only 1K pre-labeled instances to reach the performance level of the supervised models, where its performance begins to level off. While BERT-large alone can reach the levels with slightly more instances, this analysis gives a hope that UQA can serve as a strong method with no or

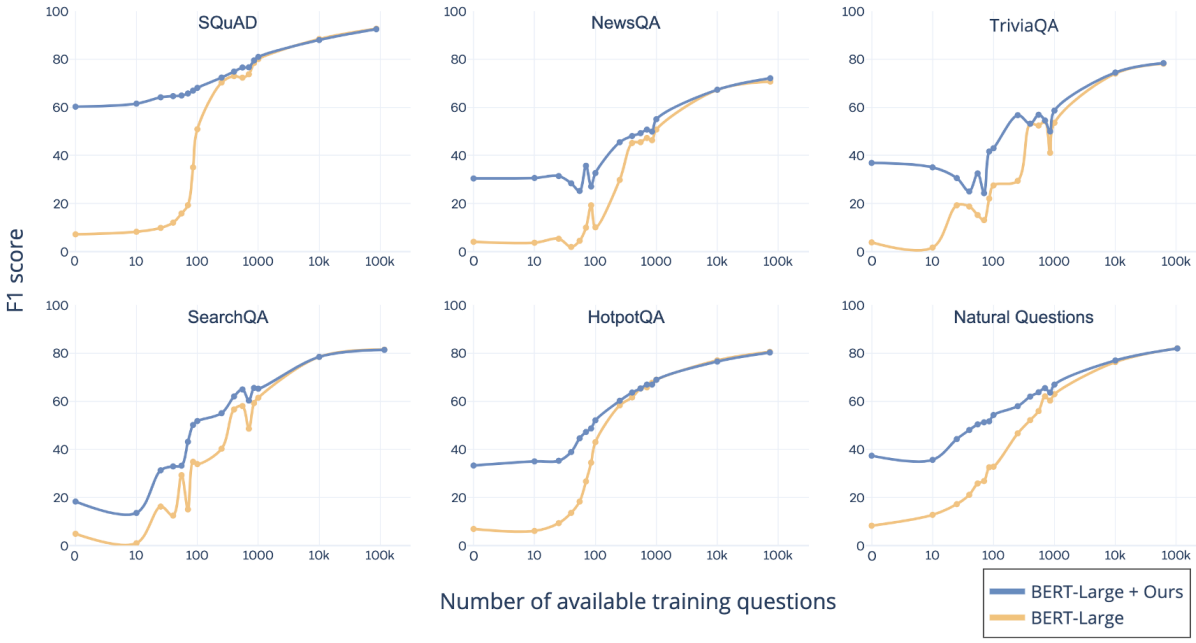

Figure 5: F1 scores on the MRQA datasets for progressively larger training dataset sizes.

a very small number of pre-labeled instances for comparable effectiveness.

4.6 Error Analysis

The analysis result in Table 4 is intended to give an insight into where our method fails. The first example shows that the question deviates from the context, perhaps due to the heavy use of LM-type. The second example suggests that our model suffers from the problem of handling pronouns. In the last example, we can see that the end of the question is not natural, indicating that our model should improve on learning how to end a question. A common problem we observe from these cases is the overuse of "you" when it is unnecessary.

Context	Generated Question
The final was a one-sided affair, with *Suburbs* proving too strong for the southerners.	Where did you get the final of this round in adelaide?
... Later he went to New College, Oxford, where he completed an *M.A.* and D.Phil. in Indian history ...	What was he doing in new college of oxford and did you study history ?
In general case, the *HJB* equation does not have a classical (smooth) solution.	What is general case of the equation in general case : " if you say that

Table 4: Examples of incorrectly generated questions using our student module. Answers in italics.

5 Related Work

Lewis et al. (2019) proposed the task of UQA and modeled it with an unsupervised question generation (UQG) task. They make use of named entities and noun phrases as answers and create "fill-in-the-blank" cloze questions as a way to achieve UQG. Their scheme is to identify a sentence that contains an answer, which is then masked to create a cloze question. Since these questions do not look natural, they use back-translation to convert cloze questions into more natural-looking ones. We argue that the resulting questions are so similar to the contexts that it is hard to train a robust QA model.

Puri et al. (2020) propose to create questions by training GPT-2 on the SQuAD training set (Rajpurkar et al., 2016) and show there is a huge performance gap between a trained QG model based on GPT-2 and the non-trained version. Klein and Nabi (2019) argue that since GPT-2 is trained for general text prediction, the result is not appropriate for the QG task. Consequently, the questions generated as such are not guaranteed to be answerable. To overcome this problem, they propose to leverage the connection between QA and QG so that GPT-2 is trained for QG, resulting in a QG-optimized GPT-2 model. Unlike these two approaches, our work uses a pre-trained version of GPT-2, which is not trained on a QG dataset, and demonstrates that we can improve the quality of questions in a completely

unsupervised way and narrow the performance gap between the supervised and unsupervised QA.

Hinton et al. (2015) propose the teacher-student architecture for knowledge distillation. This method allows compressing the knowledge of a large model, the teacher, into a smaller model, the student, which is also able to generalize the knowledge of the teacher. Unlike the original purpose of this architecture, we devise a teacher module that regularizes two question generation models, while the role of the student is to generalize this regularization process.

6 Conclusion and Future Work

We have proposed a novel method for unsupervised question generation (QG), where the questions are generated with a teacher-student architecture. The teacher is composed of two distinctive QG models as assistants and a regularization module that attempts to stay unbiased between the two styles of QG. As the main thrust of this structure, the regularization scheme takes a novel approach of selecting the next tokens in a probabilistic way when a question is generated. This knowledge is implicitly transferred to the student module so that it can mitigate the drawbacks of the two QG models in the teacher module and generate higher quality questions. To encourage further development of unsupervised question answering, we release the QA dataset generated by our student model.

With a series of experiments across the in-domain MRQA shared tasks, we demonstrate the effectiveness of the proposed method as well as its generalizability. We also provide an insight as to how the proposed method can help progress toward zero-shot and few-shot learning.

As reflected in the qualitative analysis, the current method still generates unreasonable questions that, for example, deviate too much from the context, end unnaturally, and fail to handle pronouns appropriately. To handle those cases, we need to look into what other types of QG can serve as new teachers and how the regularization needs to evolve. We leave it as future work in addition to the generalization of this approach to other generative models besides QG.

Acknowledgments

This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (2017M3C4A7065962).

References

Ying-Hong Chan and Yao-Chung Fan. 2019. A recurrent BERT-based model for question generation. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 154–162, Hong Kong, China. Association for Computational Linguistics.

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, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Matthew Dunn, Levent Sagun, Mike Higgins, V Ugur Guney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*.

Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. 2019. MRQA 2019 shared task: Evaluating generalization in reading comprehension. In *Proceedings of 2nd Machine Reading for Reading Comprehension (MRQA) Workshop at EMNLP*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Dan Hendrycks and Kevin Gimpel. 2016. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.

Junmo Kang, Haritz Puerto San Roman, and Sung-Hyon Myaeng. 2019. Let me know what to ask: Interrogative-word-aware question generation. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 163–171, Hong Kong, China. Association for Computational Linguistics.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Tassilo Klein and Moin Nabi. 2019. Learning to answer by learning to ask: Getting the best of gpt-2 and bert worlds. *arXiv preprint arXiv:1911.02365*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*.

Patrick Lewis, Ludovic Denoyer, and Sebastian Riedel. 2019. Unsupervised question answering by cloze translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4896–4910.

Raul Puri, Ryan Spring, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. 2020. Training question answering models from synthetic data. *arXiv preprint arXiv:2002.09599*.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

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*, pages 2383–2392.

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*, pages 191–200.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

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

A Appendix

A.1 Question Generation with GPT-2

The dataset used to train the LM-type QG from Section 2 and the LM-type assistant is generated using GPT-2 (Radford et al., 2019). The input to GPT-2 is the concatenation of the context, the string "A:", the answer, and the string "Q:" for a given instance. "A:" is used to let GPT-2 know that the following text is the answer, whereas "Q:" must be followed by a question, the text that the model has to generate. The question filter (Figure 4) eliminates potentially unanswerable questions. As shown in (Radford et al., 2019), if we input the context, question, and the string "A:" into GPT-2, the generated text is the answer to the question. Following this setting, we input the context and the generated question into GPT-2 to obtain a text that should serve as the answer. If this text is indeed the same as the answer that was employed to generate the question, then it passes the test of the filter.

A.2 Extended Penalized Sampling

In order to penalize repeated tokens from the output distribution of the QG modules, penalized sampling (Keskar et al., 2019) has been applied. Given a list of generated tokens g, each token is penalized by discounting with the penalty score α only if a token $w \in V$ has been previously generated. We additionally use softmax-temperature (Hinton et al., 2015) to control the smoothness of the distribution. Formally:

$$P(w|X_{t-1}) = \frac{exp(w_i)/(T \cdot I(w_i \in g))}{\sum_j exp(w_j)/(T \cdot I(w_j \in g))}$$
$$I(c) = \alpha \text{ if } c == True \text{ else } 1$$

where T denotes temperature. If T goes higher, the distribution gets smoother. For all the experiments, we set α and T to 1.5 and 1.0, respectively.

We similarly applied this technique to prevent from generating *wh*-words and the answer, which might not be useful for a question. For each time step of generating a token, we forced probabilities of answer tokens and all *wh*-words (only after a first *wh*-word is generated) to be zero.

Since LM-type QG tends to produce a shorter question, regularized questions by the teacher can be too short, resulting in meaningless questions. To address this, we controlled the length of generated tokens by penalizing some special tokens $w_s \in$ [*period (.), comma (,), quote (')*, *question mark (?)*,

[SEP]], which cause the question to be shortened.

$$P'(w_s) = \frac{P(w_s)}{L/(|g|+1)}$$

where L is a hyperparameter to control the length. The penalty becomes smaller as the number of the generated tokens gets larger. We used 15 for L during all the experiments.

Due to the computational cost, only the top-10 probabilities were used as soft target distribution to train the student module. We believe that top-10 is enough to transfer the teacher's knowledge as the probabilities except the top-10 are almost negligible.

A.3 Student Training Algorithm

Algorithm 1: Train the student using soft target distributions produced by the teacher.

1 **Input:** Contexts, pre-trained $\mathcal{D}$, pre-trained θ_{LM}, pre-trained θ_{CP}
2 **Output:** θ_{Stu}
3 Initialize θ_{Stu}
4 **foreach** *Context C* $\in$ *Contexts* **do**
5 t $\leftarrow$ 1
6 A, wh-word $\leftarrow$ Preprocessing(C)
7 $C' \leftarrow [c_1, c_2, ..., [HL], A, [HL], ..., c_{|C|}]$
8 $\hat{q}_1 \leftarrow$ wh-word
9 **while** $\hat{q}_t \neq$ *"?"* ***and*** $t \leq$ *max_len* **do**
10 t $\leftarrow$ t + 1
11 $X_{t-1} \leftarrow ([CLS], C', [SEP], \hat{q_{<t}}, [MASK])$
 `// Teacher Module`
12 type $\leftarrow \mathcal{D}(C,\ q_{<t}) \in \{LM, CP\}$
13 target $\leftarrow P(w|X_{t-1},\ \theta_{type})\ w \in V$
 `// Student Module`
14 pred $\leftarrow P(\hat{w}|X_{t-1},\ \theta_{Stu})\ w \in V$
15 $\mathcal{L}_{Stu} \leftarrow D_{KL}$(pred || target)
16 grad $\leftarrow$ backward($\mathcal{L}_{Stu}$)
17 $\theta_{Stu} \leftarrow$ update(θ_{Stu}, grad)
18 $\hat{q}_t \leftarrow \arg\max_w$(target)
19 **end**
20 **end**
21 **return** θ_{Stu}

Algorithm 1 shows the proposed pipeline system to train the student module. The inputs are the pre-trained copy-type (θ_{CP}) and LM-type (θ_{LM}) assistants, the discriminator ($\mathcal{D}$), and the contexts. The output of the pipeline system is a trained student model (θ_{Stu}). In line 6, *Preprocessing* refers to the process of outputting an answer A and a wh-word from an input context.

A.4 Detailed Hyperparameters

For all the models we used, we did not perform an exhaustive hyperparameter search. In fact, most of the hyperparameters are from Hugging Face's Transformers library [7] (Wolf et al., 2019) for BERT-QA (Devlin et al., 2019) and the discriminator schemes, or provided by Chan and Fan (2019) for BERT-HLSQG model. Although a hyperparameter tuning may achieve a performance boost, we opt for a complete in-detail analysis of our proposed method rather than attempting to achieve the highest possible performance.

QA Model We train the pre-trained *bert-large-uncased-whole-word-masking* model and fined-tuned on 10K training instances with 2 epochs and batch size of 8. We use the Adam optimizer with the learning rate of 3e-5, β_1 = 0.9, β_2 = 0.999 and epsilon of 1e-8, without a weight decay. We use a GELU (Hendrycks and Gimpel, 2016) as an activation, 384 for max sequence length, 128 for doc stride, and 1e-12 for epsilon in the layer normalization. We do not apply dropout for better analyses on other models that are part of our proposed approach. The number of parameters is 340 million.

Discriminator We train the pre-trained *bert-base-uncased* model on 70K training instances (35K instances for each question type, copy-type and LM-type) with batch size of 10 and 5 epochs. All the other hyperparameters are the same as the QA model, except for the dropout probability, which is set to 0.1 on all the layers. The number of parameters is 110 million.

QG Model The QG models are based on BERT-HLSQG. We train the models on 10K training instances with batch size of 6 and 2 epochs. The BERT-HLSQG model is based on the BERT-base, and the max length for questions is 42. All other hyperparameters are the same as the discriminator model. The number of parameters is 110M.

A.5 EM Scores on MRQA shared task datasets

The Exact Match (EM) scores of our model and the baselines in the MRQA shared tasks datasets are shown in Table 5.

[7] https://github.com/huggingface/transformers

Model	SQuAD	NewsQA	TriviaQA	SearchQA	HotpotQA	Natural Questions
Lewis et al. (2019)	45.2	19.1	26.2	**29.1**	20.9	17.7
LM-type QG	42.0	18.8	26.6	14.0	22.1	26.6
Copy-type QG	40.1	15.6	21.5	26.8	17.0	12.8
Teacher	47.5	20.1	26.6	8.5	**23.0**	26.2
Student	**49.8**	**21.2**	**30.7**	14.1	22.7	**27.2**

Table 5: EM scores of the baselines including SOTA on SQuAD and the two QG models, and our proposed approach (Teacher and Student) on the in-domain MRQA shared task datasets. The Student is our final QG model.