Unified Question Generation with Continual Lifelong Learning

Wei Yuan†
School of Information Technology and Electrical Engineering,
The University of Queensland
Brisbane, QLD, Australia
w.yuan@uq.edu.au

Hongzhi Yin†
School of Information Technology and Electrical Engineering,
The University of Queensland
Brisbane, QLD, Australia
h.yin1@uq.edu.au

Tong Chen
School of Information Technology and Electrical Engineering,
The University of Queensland
Brisbane, QLD, Australia
tong.chen@uq.edu.au

Qiu Feng Wang
Department of School of Computer Science and Engineering,
Southeast University
Nanjing, China
qfwang@seu.edu.cn

Lizhen Cui
School of Software,
Shandong University
Jinan, China
clz@sdu.edu.cn

ABSTRACT

Question Generation (QG), as a challenging Natural Language Processing task, aims at generating questions based on given answers and context. Existing QG methods mainly focus on building or training models for specific QG datasets. These works are subject to two major limitations: (1) They are dedicated to specific QG formats (e.g., answer-extraction or multi-choice QG), therefore, if we want to address a new format of QG, a re-design of the QG model is required. (2) Optimal performance is only achieved on the dataset they were just trained on. As a result, we have to train and keep various QG models for different QG datasets, which is resource-intensive and ungeneralizable.

To solve the problems, we propose a model named Unified-QG based on lifelong learning techniques, which can continually learn QG tasks across different datasets and formats. Specifically, we first build a format-convert encoding to transform different kinds of QG formats into a unified representation. Then, a method named STRIDER (Similarity Reused Difficult Example Replay) is built to alleviate catastrophic forgetting in continual QG learning. Extensive experiments were conducted on 8 QG datasets across 4 QG formats (answer-extraction, answer-abstraction, multi-choice, and boolean QG) to demonstrate the effectiveness of our approach. Experimental results demonstrate that our Unified-QG can effectively and continually adapt to QG tasks when datasets and formats vary. In addition, we verify the ability of a single trained Unified-QG model in improving 8 Question Answering (QA) systems' performance through generating synthetic QA data.

CCS CONCEPTS

• Computing methodologies → Natural language generation;
• Lifelong machine learning;
• Information systems → Question answering.

KEYWORDS

Question Generation, Question Answering, Lifelong Learning, Pre-trained Model

ACM Reference Format:
We Yuan, Hongzhi Yin, Tieke He, Tong Chen, Qiu Feng Wang, and Lizhen Cui. 2022. Unified Question Generation with Continual Lifelong Learning. In Proceedings of the ACM Web Conference 2022 (WWW ’22), April 25–29, 2022, Virtual Event, Lyon, France. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3485447.3511930

1 INTRODUCTION

Question Generation (QG), as a dual task of question answering (QA), aims at creating grammatically and semantically precise questions according to the given text. QG is an effective data enrichment technique for many web services and it plays a vital role in the improvement of contemporary web applications, such as generating questions or queries from web sources to enhance QA systems [20, 21, 71], guiding user-friendly interactions in dialogue systems [56, 63], actively enriching Frequently Asked Questions (FAQs) for web pages [36, 43], and creating educational practice exercises and assessments [39].

With advances in deep learning, deep neural network-based QG approaches have achieved remarkable progress [50]. However, most of the existing models are tailor-made for specific QG datasets and task formats. For example, recent models proposed by Zhao et al. [86], Ma et al. [45], and Yuan et al. [81] achieve state-of-the-art performance on a benchmark dataset SQuAD. However, these models are only applicable to answer-extraction QG since they require the answers to be a span of words in context. Moreover, even for the same QG format, those models trained on one dataset are unable maintain their performance on a different one due to constrained generalizability. Such inflexibility also exists in models built for other QG formats like multi-choice [9]. In real applications, there
Lifelong learning is more efficient than multitask learning because it does not require retraining on all historical data when adapting to new tasks (i.e., datasets in our case), which significantly reduces the computation and storage costs. However, lifelong learning often suffers from catastrophic forgetting [22, 47], which means a model tends to completely and abruptly forget learned knowledge when learning new information. To alleviate this issue, we propose a method called STRIDER (short for SimilariTy RegularIzed Difficult Example Replay). STRIDER stores a small set of prioritized samples from past datasets, and replays them every time when a new dataset is available. This rehearsal method has been widely used in lifelong learning and achieved great success [13, 29, 48, 61]. The main challenge of this method, especially under our QG setting, is how to select the set of prioritized data. In this paper, we propose a novel data selection scheme that chooses representative examples from historical data for QG based on difficulty.

Meanwhile, the size of selected samples should be small enough to reduce the storage cost, therefore, only rely on replaying prioritized data cannot fully avoid catastrophic forgetting. To further mitigate the forgetting problem, STRIDER incorporates a parameter update regularization method, namely Elastic Weight Consolidation (EWC) [34]. Specifically, we calculate EWC by approximating the Fisher Matrix with the chosen examples [48] to improve computational efficiency. Lastly, we point out that different QG datasets have different similarities. For example, an answer-extraction QG dataset stands a good chance to be more similar to another answer-extraction dataset, rather than a multi-choice dataset. Intuitively, for highly similar datasets, more useful knowledge from previous training steps could be preserved [31, 32]. Therefore, STRIDER dynamically adjusts the weight of EWC regularization based on the similarities between the previous and current dataset, so as to adaptively control the parameter updates.

To demonstrate the effectiveness of our approach, we conduct experiments across 8 QG datasets with 4 commonly used QG formats mentioned in Figure 1. Experimental results show that our solution, Unified-QG, can continually learn question generation across different datasets and formats. To sum up, our contributions are three-fold:

- We propose Unified-QG to continually learn question generation across datasets and formats. To the best of our knowledge, we are the first to address the generalizability issue of QG models by subsuming QG tasks under lifelong learning.
- A unified QG encoding mechanism is developed to convert four popular QG formats into the text-in/text-out form, enabling QG models to learn across formats. Meanwhile, a lifelong learning method named STRIDER is proposed to prevent QG models from catastrophic forgetting.
- Extensive experiments are conducted on 8 QG datasets with 4 QG formats to show our model’s continual learning ability. The ablation studies demonstrate the importance of each component of STRIDER. Furthermore, we employ one single Unified-QG to improve 8 Question Answering (QA) models’ performance on 8 datasets respectively, which has been infeasible for traditional dedicated QG models.

The remainder of this paper is organized as follows. Section 2 presents related research on question generation and lifelong learning. There are wide varieties of QG datasets and formats. For example, an online shopping site needs to summarize FAQs for heterogeneous types of products with different formats, and new product types can emerge from time to time. Since current QG models are only applicable to specific datasets and QG formats, we have to re-design, re-train, and store a large number of new models when handling different QG tasks on different datasets, which incurs serious computational and storage costs, reducing the usability and scalability of these models on large-scale web applications.

In light of this, we propose to construct a unified QG model which can generate questions across both datasets and formats. Nevertheless, building such a unified QG model is non-trivial. The first challenge is that different QG formats consist of different semantic input components, which require dedicated encoding schemes from QG models to process. Figure 1 shows the core components of four popular QG formats. For answer-extraction QG, the answer is a part of the passage, while in answer-abbreviation QG, the answer is not featured in the context. The multi-choice QG contains distractors that do not exist in the other three formats, and the answer of boolean QG is simply ‘yes’ or ‘no’. Existing QG methods develop specific input interfaces for these different components respectively, lacking the ability to learn across formats. Inspired by the recently released T5 model [57], we propose a unified QG encoding mechanism to transform each QG format into a unified representation, eliminating the boundaries brought by different input components.

Although unified QG encoding enables models to process question generation across formats, how to effectively and efficiently train a QG model across multiple datasets is still challenging. A straightforward solution is to use multitask learning [16], but it needs to retrain the QG model using all the historical data whenever a new dataset is available. As a result, it is not scalable due to the linearly increasing computation and storage costs. To tackle this problem, we introduce the lifelong learning concept to QG training.
learning. Section 3 provides the problem definition and illuminates the differences among lifelong, multitask learning, meta learning, and transfer learning settings. Section 4 introduces Unified-QG details. Section 5 describes the evaluation datasets and metrics, followed by results and discussions. Section 6 concludes our work.

2 RELATED WORK

2.1 Question Generation

The emergence of large-scale datasets play a crucial role in QG systems development. Since most of datasets are manually built from real web sources, QG systems can learn to enrich questions for web documents in a human-like way. In principle, as a dual task of QA, any QA datasets can be used for QG [50]. SQuAD [58], MS-MARCO [4] and newsQA [73] are three famous datasets used for answer-extraction QG, collected from Wikipedia, Bing search logs, and CNN news respectively. Unlike the previous three datasets, NarrativeQA [35] does not restrict the answers to be the span of texts in the articles, therefore, it can be used as an answer-abstraction QG dataset. Race [38], McTest [64], OpenbookQA [49], and ARC [15] are commonly used multi-choice QG datasets. BoolQA [14] is a typical boolean QG dataset, gathered from Google search engine.

Upon the variety of large-scale datasets, the QG research has made great achievements through neural network approaches. Zhou et al. [87] and Du et al. [19] are the first to explore how to address the QG problem via building end-to-end neural models. They apply a sequence-to-sequence framework with attention mechanism [3] and pointer network [23], which becomes a typical framework for later studies. To alleviate the mismatch problems w.r.t. interrogative words, Sun et al. [70] adopt an answer-aware decoding mode to generate the question words given the answer’s hidden states. To widen the input context, Zhao et al. [86] employ gated self-attention with a maxout pointer to enhance the model’s long sentence processing ability. Yuan et al. [81] enrich the input information by fusing deep linguistic information based on BERT and achieve state-of-the-art performance. However, most of these works focus on answer-extraction QG, and cannot address other QG formats, due to the assumption that the answer is a span of texts in the context. Some works train QG models with user-written QA pairs [10, 30, 76, 80], which are highly relevant to answer-abstraction QG. Boolean and multi-choice QG are partially similar, where the prior one can be viewed as two-option styled multi-choice QG [9, 39, 62]. Furthermore, apart from based on text documents, many QG works attempt to generate questions with other kinds of web materials, such as knowledge graph (KG) [79] and tables [89].

Recently, applying large-scale pretrained models in QG attracts more and more researchers’ interests. Chan et al. build a recurrent BERT to output one question word at a recurrent step [11, 12], but it is time-consuming. The generative pretrained models such as UNILM [18], T5 [57], PEGASUS [82], and UNILMV2 [5] report the model’s QG scores finetuned on SQuAD [58] dataset, but they do not explore the idea of building a unified QG.

In this paper, we employ the pretrained model T5 as the skeleton of our Unified-QG, and conduct experiments on 8 QG benchmarks (SQuAD, NarrativeQA, RACE, McTest, OpenbookQA, ARC-easy, ARC-hard, and BoolQA) with 4 common formats (extraction, abstraction, multi-choice, and boolean QG).

2.2 Lifelong Learning

Lifelong learning (also referred to as continual learning) is a type of machine learning paradigm, which aims to continually learn across time and does not severely forget previously learned tasks [53]. Since it enables models to expand their knowledge to new domains or functionalities incrementally, lifelong learning has been widely used in artificial intelligent web applications [24, 25, 28, 75, 77, 84]. The main challenge in lifelong learning is that models incline to catastrophically forget existing knowledge when learning from novel tasks [72].

Generally, approaches to alleviate catastrophic forgetting can be categorized into three families: rehearsal, regularization, and architectural methods. Architectural methods attempt to dynamically apply modular changes or add task-specific parameters to prevent forgetting [46]. However, the architectural method’s parameters will dramatically increase when the number of tasks grows. Rehearsal methods mitigate catastrophic forgetting by retaining some training examples and replay them later. Therefore, how to choose appropriate examples is the key challenge. Rebuffi et al. [61] propose iCaRL which selects training data using Herding techniques [78]. Ramalho et al. [59] collect less confident examples and propose iCaRL which selects training data using Herding techniques [78]. Ramalho et al. [59] collect less confident examples and propose iCaRL which selects training data using Herding techniques [78]. Ramalho et al. [59] collect less confident examples and propose iCaRL which selects training data using Herding techniques [78]. Ramalho et al. [59] collect less confident examples and propose iCaRL which selects training data using Herding techniques [78]. Ramalho et al. [59] collect less confident examples and propose iCaRL which selects training data using Herding techniques [78]. Ramalho et al. [59] collect less confident examples and propose iCaRL which selects training data using Herding techniques [78].
works rectify models’ parameter updates according to knowledge distillation [8, 27, 40, 85].

In this paper, we present STRIDER, which incorporates both rehearsal and regularization to avoid catastrophic forgetting.

3 PROBLEM DESCRIPTION
3.1 Question Generation
Generally, the goal of QG is to generate a question \( Q \) given a context \( C \) and an answer \( A \). Formally:

\[
\hat{Q} = \text{argmax } P(Q|C,A)
\]

The context \( C \), answer \( A \), and question \( Q \) are composed of a list of words. Eq. 1 is a unified formal statement of QG. With different assumptions of \( C, A, \) and \( Q \), QG can be classified into different types. For answer-extraction QG, answer \( A \) is a subspan of \( C \). While for answer-abstraction QG, answer \( A \) can be any novel words. For multi-choice QG, the context not only includes description, but also the distractors. For boolean QG, the word in an answer is restricted to the binary “yes” or “no”. The QG dataset \( D \) consists of the question-answer-context triples \( D = \{(c_i, a_i, q_i)\}_{i=0}^{n} \).

3.2 Lifelong Learning of QG
As mentioned before, current QG models are dedicated to specific datasets with specific formats, and few research studies QG across datasets and formats. In this paper, we implement such QG by using lifelong learning strategies. Following previous lifelong learning works’ setting [34, 48, 61], we assume each QG dataset represents a QG task and the new QG data arrives task by task. Let \( f_t \) be the well-trained QG model in task \( t \). The new dataset \( D_{t+1} \) for task \( t + 1 \) is used to update model \( f_t \). After updating, the newest model \( f_{t+1} \) should have good performance in all learned tasks \( 1 : t + 1 \).

Different from multitask learning, lifelong learning does not require retraining the model \( f_t \) on all the datasets \( D_{1:t+1} \) when a new task \( t + 1 \) comes in. Figure 2 illustrates the differences among traditional QG, QG with multitask learning, and QG with lifelong learning. Traditional QG models need to be specifically designed for each task. Multitask learning QG should be retrained on all previous datasets when a new task arrives, while lifelong learning QG only needs to be continually trained on the new dataset and will perform well in all presented tasks.

It is also important to point out the setting’s differences among lifelong learning, transfer learning and meta learning. Transfer learning aims to transfer the knowledge from the source task to the target task in order to improve model performance on the target task. It does not consider the performance of previous tasks [6, 54]. On the other hand, meta-learning focuses on learning generic knowledge from a small set of data and numerous tasks, so that it can quickly adapt to a new task. Therefore, it enhances the learning ability without considering performance collapse for past tasks.

In the experiment section, we will discuss the superiority of our lifelong learning method for Unified-QG, compared with transfer learning and multitask learning.

4 UNIFIED QG WITH LIFELONG LEARNING
4.1 Overview of Unified QG
We first present an overview of our Unified-QG from two aspects: the model architecture and the lifelong learning method STRIDER, which are depicted in Figure 3. For the model architecture, we propose a unified QG encoding method to convert different QG formats into a unified form, endowing QG models with the ability to handle diversified formats. Meanwhile, we adopt T5 [57] as the backbone of our QG model. T5 is instructed to perform particular tasks by adding prefixes. During continual training, it will identify the prefix meaning we added in the unified QG encoding. Although the modified QG model can now produce questions across formats with unified QG encoding, it still suffers from the forgetting problem. Hence, we propose STRIDER to alleviate this problem. STRIDER addresses forgetting by retaining and replaying some challenging and representative examples of historical data, and elastically regularizes the model’s update.

4.2 Model Architecture
4.2.1 Unified QG Encoding. As shown in Figure 1, different QG formats contain different components. Traditional QG models need specific input interfaces to handle certain formats. In order to build a unified QG that can work across formats, we propose an encoding scheme for QG instances to unify various formats.

The multi-format encoding is inspired by T5 model’s text-to-text encoding method. In the unified QG encoding, all the components are concatenated as a long paragraph. The concatenated components have an order of answer, passage, and distractor and other components if necessary, where each component starts with a prefix. The concatenated paragraph is then fed into our Unified-QG model and outputs the question. Figure 4 shows the form of our unified QG encoding with four representation examples of different QG formats. We emphasize that although we mainly conduct experiments with these four kinds of QG formats, our unified QG encoding can easily generalize to other QG formats. After unified encoding, the QG dataset can be represented as \( D = \{(x_i, y_i)\}_{i=0}^{n} \). \( x_i \) is the concatenated input in the Figure 4 and \( y_i \) is the target question. \( x_i \) is then fed into the skeleton of QG model, T5, which will be introduced in the following part.

4.2.2 T5 for QG. We incorporate T5 [57] as the skeleton of our QG model, which is a variant of the encoder-decoder transformer model [74] pretrained in multiple tasks. It allows for different input components specified by the prefixes in the input. Its encoder
Unified Question Generation with Continual Lifelong Learning

where \( \text{LayerNorm} \) retains a small set of examples from the past training data and replay to avoid catastrophic forgetting during continual learning. We propose a similarity-based adaptive regularization method for QG named STRIDER. STRIDER is featured with similarity-based adaptive regularization and difficult data replay.

The structure of the decoder is much similar to the encoder but it includes attention mechanism after each self-attention, in order to attend to the encoder’s output. Besides, the decoder’s self-attention includes attention mechanism after each self-attention, in order to attend to the encoder’s output.

4.3 Similarity Regularized Difficult Example Replay (STRIDER)

QG models equipped with the unified QG encoding are able to produce questions across formats. However, it is subject to catastrophic forgetting, if only updated based on the new dataset/task. To alleviate such forgetting, we propose a novel lifelong learning approach for QG named STRIDER. STRIDER is featured with similarity-based adaptive regularization and difficult data replay.

**4.3.1 Difficult Example Replay.** To avoid catastrophically forgetting the learned knowledge from previous datasets, we propose to retain a small set of examples from the past training data and replay these examples in later training tasks. Such replay methods have been successful in many lifelong learning works [13, 29, 48, 61].

The effectiveness of replay methods largely depends on the quality of selected examples. In this paper, we propose a difficult example selection method to select informative and diverse data from previous datasets. The intuition is that, difficult training examples are more informative and helpful for improving the current model. Hence, with the limited replay size, we can only retain a small set of difficult historical examples for replay.

Our difficult example selection criterion on task \( t \) is as follows:

\[
S_t(x^t_i, y^t_i) = \frac{CE(x^t_i, y^t_i|\theta_t)}{|y^t_i|} \quad (3)
\]

where \( x^t_i \) and \( y^t_i \) is the \( i \)-th pair of training data in \( D_t \). \( CE(\cdot|\theta_t) \) is the cross-entropy function calculated based on model parameter \( \theta_t \), which can be interpreted as the model’s training loss on the \( i \)-th sample of dataset \( D_t \). We observe that long sentences tend to have higher accumulated loss values, therefore, we use the inverse of \( y^t_i \)’s length as a normalization factor to eliminate the impact of length on loss values. Essentially, \( S(.) \) quantifies the confidence of the model when performing QG for every given \( x^t_i \), where a higher value indicates that the selected training sample is more challenging for the model to learn. We collect the \((x^t_i, y^t_i)\) pairs that can maximize \( S(x, y) \) as the difficult example set from \( D_t \):

\[
E_t = \arg\max_{E_t \subseteq D_t \land |E_t|=N} \sum_{(x,y) \in E_t} S_t(x,y) \quad (4)
\]

where Eq. 4 aims to select \( N \) examples of \((x_t, y_t)\) pairs that can maximize the selection criterion.

Finally, during the continual training of the unified QG model \( f_t \) on dataset \( D_t \), the selected example sets from previous datasets \( E_{1:t-1} = \{E_i\}_{i=1}^{t-1} \) are integrated to augment \( D_t \). The training objective is to minimize the following function:

\[
L_R(\theta_t) = \sum_{(x,y) \in D_{1:t} \cup E_{1:t-1}} CE(x,y|\theta_t) \quad (5)
\]

4.3.2 Similarity Regularization. The size of retained historical examples should be as small as possible to reduce storage overhead such that \( |E_{1:t}| \ll |D_{1:t}| \). With a small \( N \), the effect of replaying difficult examples on alleviating catastrophic forgetting is largely restricted due to insufficient samples for each task. To further mitigate forgetting during continual learning, we propose a similarity-based adaptive regularization method Similarity Regularization.

Similarity regularization utilizes the well-established Elastic Weight Consolidation (EWC) [34] as the main regularization term. Inspired by the idea that synaptic consolidation achieves continual learning via reducing the plasticity of synapses in human brains, EWC imposes restrictions on parameters that are important for historical tasks:

\[
L_E(\theta_t) = \sum_i \lambda F_i(\theta_t, \theta_{t-1})^2 \quad (6)
\]

where \( F_i = \nabla^2 CE(x,y|\theta_{t-1}) \) s.t. \((x,y) \in E_{1:t-1}\) denotes the \( i \)-th element in the diagonal Fisher Information Matrix. The Fisher Matrix describes the covariance of gradients, which is a measure for the importance of parameters. Note that we calculate \( F_i \) only with the example sets, without the need for storing the whole previous
data. \( \lambda \) controls the influence of the EWC term. As shown in Eq. 6, EWC elastically slows the changes of parameters that are important for historical tasks, and give the “unimportant” parameters more plasticity.

In QG, we observe that the similarities among datasets vary significantly. For example, datasets that belong to answer-extraction QG (e.g. SQuAD) might be more similar to answer-abstraction QG datasets (e.g. NarrativeQA) than to multi-choice QG datasets (e.g. RACE). Naturally, the model will need less parameter updates on the dataset that is similar to previous data than on the dissimilar one. Therefore, we propose a similarity-based adaptive weight \( \lambda \) to control EWC regularization. To approximate this similarity, we treat the current dataset and previous data as two documents and vectorize them by using TF-IDF. TF-IDF is often used to measure the relevance between texts \([55, 60, 69]\), and is parameter-free and highly efficient to compute compared with other alternatives. Then, we employ cosine similarity to calculate the adaptive weight \( \lambda \):

\[
\lambda = \lambda_{ori} \cdot \frac{T(D_t) \cdot T(E_{t-1})}{\|T(D_t)\| \cdot \|T(E_{t-1})\|} \quad (7)
\]

where \( \lambda_{ori} \) is a hyper-parameter that represents the original weight of EWC, and \( T(\cdot) \) is the TF-IDF function that vectorizes a dataset. Note that we use the selected example sets to represent the historical data instead of using all the historical data. In general, \( \lambda \) increases when the current set is similar to historical examples, thus reducing the degree of parameter updating on task \( t \).

5 EXPERIMENTS

5.1 Datasets

We select 8 representative datasets that cover 4 QG formats as our evaluation datasets, including:

- Answer-extraction QG: SQuADv1.1 [58]
- Answer-abstraction QG: NarrativeQA [35]
- Multi-choice QG: RACE [38], McTest [64], OpenbookQA [49], ARC-easy, ARC-hard [15]
- Boolean QG: BoolQA [14]

We assume datasets arrive in the following order: "McTest→SQuAD→RACE→NarrativeQA→Arc-easy→Arc-hard→OpenbookQA→BoolQA", which corresponds to the exact release dates of these datasets in the real world. Details on dataset characteristics, statistics, and splitting strategies are in Appendix A.1.

5.2 Evaluation Metrics

Following existing QG works \([81, 88]\), we adopt the widely used n-gram based automatic evaluation metrics, BLEU-1–4 \([51]\), METEOR \([17]\), and ROUGE-L \([41]\), to evaluate our models. BLEU measures the prediction quality based on the co-occurrence n-gram frequency between references and predictions \([7]\). BLEU-1 measures the precision of unigram, while higher-order BLEU introduces a bias towards sentence fluency \([67]\). METEOR estimates candidate text’s quality by calculating the harmonic mean of unigram precision and recall based on not only exact word matching, but also stemming and synonym matching. ROUGE-L is the Longest Common Subsequence (LCS) based ROUGE \([42]\). To some extent, it takes into account the sentence-level structure similarity between candidate text and reference and can identify the longest co-occurring n-gram. In the following tables, we use “B1~4”, “M”, and “RL” represent BLEU-1~4, METEOR, and ROUGE-L.

Since none of the existing work has studied QG across datasets and formats by using lifelong learning techniques, we employ two additional metrics to better evaluate the lifelong learning ability:

\[
M_{\text{seen}} = \frac{1}{T} \sum_{i=1}^{T} M_{\text{seen},i} \quad M_{\text{seen},i} = \frac{1}{i} \sum_{i=1}^{t} M_{i,i} \quad (8)
\]

\[
M_{\text{first}} = \frac{1}{T} \sum_{i=1}^{T} M_{\text{first},i} \quad M_{\text{first},i} = M_{1,i} \quad (9)
\]

where \( M_{\text{seen},i} \) denotes the model’s average performance on all seen tasks after the task \( i \) has been learned. \( M_{i,i} \) represents the model’s performance on \( t \)-th task after learning task \( i \). \( M_{\text{seen}} \) is the average of the model’s \( M_{\text{seen},i} \) on all tasks. \( M_{\text{first}} \) is the performance on the first task after learning \( i \)-th task, which is equivalent to \( M_{1,i} \). \( M_{\text{first}} \) denotes the average performance of the model on the first task at the end of learning each task. \( M \) can be any of the evaluation metrics mentioned above. \( M_{\text{seen}} \) evaluates model’s overall performance on historical tasks, while \( M_{\text{first}} \) reflects the ability to avoid catastrophic forgetting.

5.3 Baseline Methods

As we mentioned in Section 1 and Section 3.2, none of the existing QG models can create questions across datasets and formats, since they are dedicated to specific tasks. For the purposes of comparison, we train a Multitask-QG, a Finetuned-QG, and a Random-selected QG as our baseline. All of the baselines utilize the same model architecture and QG encoding as our Unified-QG.

For Multitask-QG, the model is trained in a multi-task way. That is to say, the QG model is trained on a current dataset and all historical data, for each new task. It can achieve the "upper-bound" performance of overall continual lifelong learning since it preserves and is trained on all current and historical data.

For Finetuned-QG, the model is trained in a transfer learning style. With the progress of the tasks, the model is initialized with the checkpoint obtained from the last task, and then, it is finetuned with data from the current task. This method might perform well on the current task but will suffer catastrophic forgetting.

Random-selected QG selects the examples from historical data uniformly. The advantage of the random-selected scheme is that it is simple and it can keep the selected samples and original data the same distribution to a large extent.

Note that all of the baseline methods utilize our proposed unified QG encoding method so that they can work across formats. And all of the hyper-parameters settings for these baselines are the same as our Unified-QG. The detailed implementation is in Appendix A.2

5.4 Overall Evaluation Results

The top half of Table 1 presents the results in terms of \( M_{\text{seen}} \) which reflects the overall performance on all the datasets. Finetuned-QG has the worst performance among all the comparison methods, since it suffers from the catastrophic forgetting problem. Random-selected QG also has relatively poor performance, which shows the superiority of our difficulty-based sampling method. As mentioned in Section 5.3, Multitask-QG can be viewed as the "upper-bound"
with regarding to $M_{seen}$, since it stores all historical data and costs much more GPU computations to train the model. Our Unified-QG achieves very close performance to the Multitask-QG, with much less computing and storage resources.

Table 1: The comparison of each model’s average continual learning performance ($M_{seen}$ or $M_{first}$). $^*$ means the “upper-bound” under the metrics $M_{seen}$.

| Model               | $B1$  | $B2$  | $B3$  | $B4$  | $M$   | $RL$ |
|---------------------|-------|-------|-------|-------|-------|------|
| Finetuned-QG        | 32.58 | 20.04 | 13.84 | 10.02 | 17.83 | 34.39|
| Random-selected QG  | 40.93 | 26.71 | 19.18 | 14.34 | 20.66 | 41.20|
| Multitask-QG*       | 42.31 | 28.31 | 20.70 | 15.57 | 21.56 | 42.85|
| Unified-QG (ours)   | 42.15 | 28.17 | 20.58 | 15.69 | 21.49 | 42.79|

$M_{first}$

| Model               | $B1$  | $B2$  | $B3$  | $B4$  | $M$   | $RL$ |
|---------------------|-------|-------|-------|-------|-------|------|
| Finetuned-QG        | 34.97 | 21.37 | 14.36 | 10.27 | 20.02 | 37.62|
| Random-selected QG  | 48.63 | 33.78 | 25.83 | 20.40 | 24.26 | 48.92|
| Multitask-QG        | 47.84 | 33.72 | 25.89 | 20.36 | 24.32 | 48.76|
| Unified-QG (ours)   | 51.22 | 36.61 | 28.42 | 22.54 | 25.92 | 51.79|

(a) The trend of average BLEU-4 scores on all seen datasets.
(b) The trend of BLEU-4 scores on the first dataset.

Figure 5: Detailed performance with $M_{seen}$ and $M_{first}$.

Figure 5a further shows the trend of BLEU-4 scores on all seen datasets (BLEU-4$_{seen}$). We can observe that all the methods achieve comparable performance on the very front tasks, but the finetune method falls far behind the other methods from the third task due to its catastrophic forgetting nature. The multitask method and random-selected method show similar performance, in fact, the latter one can be viewed as simplified version of the multitask method. Our Unified-QG achieves better performance after learning 4 tasks, exhibiting the strong ability to continually learn QG tasks. The reason that Multitask-QG underperforms Unified-QG in the last four datasets is that it suffers from the significant discrepancy in data sizes. Table 2 further shows that the final Multitask-QG only achieves good performance on the largest dataset RACE, and consistently underperforms on all small datasets.

5.5 Effectiveness of Overcoming Catastrophic Forgetting Problem

As we mentioned in Section 5.2, $M_{first}$ measures the effectiveness of overcoming the forgetting problem. The bottom half of Table 1 shows each approach’s average performance on the first dataset (McTestQA), which reflects the effectiveness of mitigating catastrophic forgetting. As illustrated in Table 1, simply finetune the QG model on each new dataset leads to severe forgetting, in other words, it indicates that catastrophic forgetting exists in continual learning QG. Randomly select examples from the previous data slightly alleviates forgetting but still obtains relatively poor performance. Multitask-QG performs best among all baselines, however, it utilizes all historical data when training on a new task, costing much more computation and storage resources and much longer training time. Our Unified-QG outperforms Multitask-QG significantly and consistently in terms of both effectiveness and efficiency.

Table 2: The detail performance of the last model on all the datasets. At each cell, the first line of value is for Finetuned-QG, the second line is for Random-selected QG, the third line is for Multitask-QG, and the last line is for our method. The best value is in bold.

| Dataset         | $B1$  | $B2$  | $B3$  | $B4$  | $M$   | $RL$ |
|-----------------|-------|-------|-------|-------|-------|------|
| McTest          | 31.41 | 15.76 | 9.41  | 5.82  | 17.43 | 30.51|
|                 | 50.21 | 34.83 | 26.69 | 20.94 | 24.70 | 50.35|
|                 | 48.59 | 34.92 | 26.77 | 20.96 | 24.87 | 48.76|
|                 | 54.35 | 40.08 | 32.09 | 25.48 | 27.75 | 53.54|
| SQuAD           | 23.59 | 8.68  | 4.02  | 2.058 | 11.56 | 23.02|
|                 | 44.66 | 29.13 | 20.94 | 15.61 | 22.41 | 43.91|
|                 | 45.64 | 30.10 | 21.75 | 16.30 | 23.06 | 45.46|
|                 | 46.50 | 32.29 | 22.94 | 19.20 | 23.80 | 47.42|
| RACE            | 16.35 | 6.24  | 3.00  | 1.63  | 8.58  | 14.23|
|                 | 36.51 | 24.08 | 16.58 | 11.69 | 18.75 | 32.57|
|                 | 40.71 | 28.28 | 20.36 | 15.00 | 21.34 | 37.09|
|                 | 36.59 | 24.86 | 17.11 | 12.95 | 19.33 | 33.53|
| NarrativeQA     | 20.59 | 5.16  | 1.89  | 0.86  | 9.30  | 20.30|
|                 | 37.84 | 21.23 | 14.02 | 9.83  | 17.51 | 38.47|
|                 | 38.57 | 22.13 | 14.81 | 10.51 | 18.07 | 39.37|
|                 | 38.64 | 23.45 | 16.43 | 12.19 | 18.64 | 41.59|
| Arc-easy        | 16.56 | 8.54  | 5.03  | 3.13  | 10.62 | 24.36|
|                 | 31.59 | 19.61 | 13.34 | 9.50  | 17.32 | 35.35|
|                 | 29.01 | 17.81 | 11.98 | 8.43  | 16.33 | 34.14|
|                 | 30.85 | 19.81 | 13.85 | 10.17 | 17.78 | 36.79|
| Arc-hard        | 12.05 | 5.74  | 3.09  | 1.78  | 9.10  | 21.94|
|                 | 28.18 | 16.28 | 10.21 | 6.87  | 15.75 | 31.82|
|                 | 25.33 | 14.63 | 9.34  | 6.39  | 14.57 | 31.47|
|                 | 27.04 | 16.51 | 11.05 | 7.83  | 15.82 | 33.66|
| OpenbookQA      | 22.13 | 10.93 | 5.79  | 3.22  | 11.80 | 20.67|
|                 | 30.83 | 18.21 | 11.47 | 7.62  | 18.81 | 30.50|
|                 | 35.08 | 21.68 | 14.21 | 9.73  | 19.38 | 33.34|
|                 | 34.32 | 22.57 | 15.76 | 11.33 | 19.43 | 35.18|
| BoolQA          | 49.49 | 33.70 | 24.71 | 18.51 | 23.68 | 47.04|
|                 | 47.79 | 31.86 | 23.06 | 17.14 | 22.58 | 44.96|
|                 | 47.90 | 31.62 | 22.74 | 16.73 | 22.25 | 45.16|
|                 | 48.84 | 33.10 | 24.19 | 18.04 | 23.27 | 46.08|
Table 3: The comparison of Unified-QG and 8 dedicated trained QG models.

| Dataset    | B4 Dedicated-QG | B4 Unified-QG |
|------------|----------------|--------------|
| McTest     | 24.43          | 25.48        |
| SQuAD      | 17.59          | 19.20        |
| RACE       | 17.47          | 12.95        |
| NarrativeQA| 11.05          | 12.19        |
| Arc-easy   | 9.44           | 10.17        |
| Arc-hard   | 7.49           | 7.83         |
| OpenbookQA | 13.86          | 11.33        |
| BoolQA     | 22.32          | 18.04        |

Table 2 describes the last model’s performance on each dataset. In each cell, the four values represent the performance of Finetuned-QG, Random-selected QG, Multitask-QG, and Unified-QG respectively. From Table 2 we can find that: (1) If we simply finetune the QG model task by task, at the end of training, the last model nearly forgets all the learned knowledge from previous tasks and only performs well on the last dataset. That means, in order to achieve good performance on all previous datasets using the traditional finetuning method, we have to retain all the historical model checkpoints, which will cost enormous storage. (2) Compared to Finetuned-QG, Random-selected QG can alleviate the forgetting issue to a limited extent. (3) Multitask-QG costs more computation time and storage than our approach, however, it only has better performance on RACE. (4) Our Unified-QG outperforms all of the baselines on six datasets with more efficient computation and storage.

5.6 Unified-QG vs. Dedicated-QG

We further compare our Unified-QG with dedicated trained QG models, i.e. for each dataset, we train a QG model. The architecture and all the training hyper-parameters of these dedicated QG models are the same as our Unified-QG.

Table 3 reports the BLEU-4 results of these dedicated QG models and our Unified-QG. Note that the “Dedicated-QG” does not refer to a single model, it is a generic term for 8 QG models, each of which is well-trained on only one specific dataset. Our Unified-QG outperforms the specifically trained QG on McTest, SQuAD, NarrativeQA, Arc-easy, and Arc-hard. Moreover, combined with Table 5 we can observe that our Unified-QG outperforms dedicated QG on most small-scale datasets, which means low resource QG tasks are benefitted from such continually learning.

5.7 Ablation Study

In Table 4, to understand the effects of different components, we compare STRIDER with several simplified versions, including without similarity-based adaptation (-ST), without difficulty-based example selection (-D), i.e. random selection, and without example replay (-ER). We can observe that (1) example replay is most important. Without it, the performance drops significantly from 15.69 and 22.54 to 11.24 and 11.64 in terms of BLEU-4seen and BLEU-4first; (2) difficult example replay is more effective than uniformly selected examples; (3) similarity-based regularization is also beneficial since the performance of STRIDER degrades without it.

Table 4: Ablation study for STRIDER.

| STRIDER          | B4seen | B4first |
|------------------|--------|---------|
| -ST              | 15.53  | 21.92   |
| -D               | 14.34  | 20.40   |
| -ER              | 11.24  | 11.64   |

In addition to the above evaluations, we also show Unified-QG improves 8 QA systems’ performance in Appendix B.

6 CONCLUSION

In this paper, we propose Unified-QG, which can continually learn QG tasks across datasets and formats based on lifelong learning. Specifically, our Unified-QG contains a T5-based unified QG model and a lifelong learning strategy STRIDER. The T5-based unified QG model consists of a unified QG converting mechanism that converts multiple QG formats into text-in-text-out format, and a T5 model that processes the unified input. The STRIDER includes a difficulty-based example replay and a similarity-based adaptive regularization to enable the model to continually learn how to produce questions. To the best of our knowledge, it is the first time to construct a QG model simultaneously address different QG problems crossing format boundaries based on lifelong learning approaches. We conduct extensive experiments and analyses on 8 QG datasets with 4 QG formats to demonstrate the effectiveness of our Unified-QG. Finally, we apply our single Unified-QG to improve 8 QA systems without any architectural change, which is infeasible for traditional QG models.

7 ACKNOWLEDGMENTS

The work is partly supported by the National Key Research and Development Program of China (2021YFB1715600), Australian Research Council Future Fellowship (FT210100624) and Discovery Project (DP190101985).

REFERENCES

[1] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. 2018. Memory aware synapses: Learning what (not) to forget. In Proceedings of the European Conference on Computer Vision (ECCV). 139–154.
[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. arXiv preprint arXiv:1607.06450 (2016).
[3] Dmitriy Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014).
[4] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. 2016. Ms marco: A human generated machine reading comprehension dataset. arXiv preprint arXiv:1611.09268 (2016).
[5] Hangbo Bao, Li Dong, Furui Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Jianfeng Gao, Songhao Piao, Ming Zhou, et al. 2020. Unilmv2: Pseudo-masked language models for unified language model pre-training. In International Conference on Machine Learning. PMLR, 642–652.
[6] Magdalena Biesialka, Katarzyna Biesialka, and Marta R Costa-jussà. 2020. Continual lifelong learning in natural language processing: A survey. arXiv preprint arXiv:2012.09823 (2020).
[7] Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluating the role of BLEU in machine translation research. In 11th Conference of the European Chapter of the Association for Computational Linguistics.
[8] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guíl, Cordelia Schmid, and Kartikey Alahari. 2018. End-to-end incremental learning. In Proceedings of the European conference on computer vision (ECCV). 233–248.
Unified Question Generation with Continuous Lifelong Learning

WWW ’22, April 25–29, 2022, Virtual Event, Lyon, France
We split the dataset into training, dev and testing sets with sizes 4,955, 500 and 500 respectively. Arc-easy and Arc-hard [15] questions as well as multi-choice datasets. RACE [38] collects nearly 480,000 questions and it provides an "open-book" of about 1,300 science questions and it provides an "open-book" of about 1,300 science questions. The size of training, dev, and testing sets contain 87,599, 5,280, and 5,285 examples respectively. NarrativeQA [35] is a collection of about 1,500 stories and movie scripts with summaries. About 47,000 question and answer pairs are created by crowd-source workers. The dataset is split into training, dev, and testing sets with the size of 65,494, 6,922, and 21,114, respectively.

RACE, McTest, OpenbookQA, ARC-easy, and ARC-hard are multi-choice datasets. RACE [38] collects nearly 98,000 multi-choice questions on 28,000 articles from middle and high school English tests. Since tests are constructed by experts, the questions are more challenging and often rely on multiple sentences. In this paper, we divide RACE into 87, 866, 4,887, and 4,930 sets for training, dev, and testing. McTest [64] comprises of about 500 fictional stories and 2,000 questions formed in multi-choice. Most of the questions are open-domain with less restriction on what can be asked. The size of training, dev, and testing sets for McTest are 1, 480, 160, and 160, respectively. OpenbookQA [49] contains about 6,000 open-domain science questions and it provides an "open-book" of about 1,300 science facts. Since the question is associated with some common-sense knowledge, generating such questions are more challenging. We split the dataset into training, dev, and testing sets with sizes 4,955, 500 and 500 respectively. Arc-easy and Arc-hard [15] questions are derived from ARC multi-choice question set released as part of the AI2 Reasoning Challenge in 2018 [1]. ARC provides about 8,000 questions as well as 14 million sentences about science that are related to the questions. For Arc-easy, the training, dev, and testing sets contain 2, 251, 570, and 2,376 examples respectively. And for Arc-hard, there are 1,119, 299, and 1,172 examples in the training, dev and testing sets respectively.

Table 5 presents the statistics of the above datasets, including data splitting size, the average input length (our unified QG encoding’s input length), and the average length of output (questions).

SQuAD is one of the most commonly used answer-extraction QG datasets. There are two versions, and we use SQuADv1.1 [58] in our experiment, which contains 536 Wikipedia articles with more than 100k questions. Following [81], we split the dataset into training, dev, and testing sets, each set containing 87, 599, 5,280, and 5,285 elements. NarrativeQA [35] is a collection of about 1,500 stories and movie scripts with summaries. About 47,000 question and answer pairs are created by crowd-source workers. The dataset is split into training, dev, and testing sets with the size of 65,494, 6,922, and 21,114, respectively.

RACE, McTest, OpenbookQA, ARC-easy, and ARC-hard are multi-choice datasets. RACE [38] collects nearly 98,000 multi-choice questions on 28,000 articles from middle and high school English tests. Since tests are constructed by experts, the questions are more challenging and often rely on multiple sentences. In this paper, we divide RACE into 87, 866, 4,887, and 4,930 sets for training, dev, and testing. McTest [64] comprises of about 500 fictional stories and 2,000 questions formed in multi-choice. Most of the questions are open-domain with less restriction on what can be asked. The size of training, dev, and testing sets for McTest are 1, 480, 160, and 160, respectively. OpenbookQA [49] contains about 6,000 open-domain science questions and it provides an "open-book" of about 1,300 science facts. Since the question is associated with some common-sense knowledge, generating such questions are more challenging. We split the dataset into training, dev, and testing sets with sizes 4,955, 500 and 500 respectively. Arc-easy and Arc-hard [15] questions are derived from ARC multi-choice question set released as part of the AI2 Reasoning Challenge in 2018 [1]. ARC provides about 8,000 questions as well as 14 million sentences about science that are related to the questions. For Arc-easy, the training, dev, and testing sets contain 2, 251, 570, and 2,376 examples respectively. And for Arc-hard, there are 1,119, 299, and 1,172 examples in the training, dev and testing sets respectively.

SQuAD is one of the most commonly used answer-extraction QG datasets. There are two versions, and we use SQuADv1.1 [58] in our experiment, which contains 536 Wikipedia articles with more than 100k questions. Following [81], we split the dataset into training, dev, and testing sets, each set containing 87, 599, 5,280, and 5,285 elements. NarrativeQA [35] is a collection of about 1,500 stories and movie scripts with summaries. About 47,000 question and answer pairs are created by crowd-source workers. The dataset is split into training, dev, and testing sets with the size of 65,494, 6,922, and 21,114, respectively.

RACE, McTest, OpenbookQA, ARC-easy, and ARC-hard are multi-choice datasets. RACE [38] collects nearly 98,000 multi-choice questions on 28,000 articles from middle and high school English tests. Since tests are constructed by experts, the questions are more challenging and often rely on multiple sentences. In this paper, we divide RACE into 87, 866, 4,887, and 4,930 sets for training, dev, and testing. McTest [64] comprises of about 500 fictional stories and 2,000 questions formed in multi-choice. Most of the questions are open-domain with less restriction on what can be asked. The size of training, dev, and testing sets for McTest are 1, 480, 160, and 160, respectively. OpenbookQA [49] contains about 6,000 open-domain science questions and it provides an "open-book" of about 1,300 science facts. Since the question is associated with some common-sense knowledge, generating such questions are more challenging. We split the dataset into training, dev, and testing sets with sizes 4,955, 500 and 500 respectively. Arc-easy and Arc-hard [15] questions are derived from ARC multi-choice question set released as part of the AI2 Reasoning Challenge in 2018 [1]. ARC provides about 8,000 questions as well as 14 million sentences about science that are related to the questions. For Arc-easy, the training, dev, and testing sets contain 2, 251, 570, and 2,376 examples respectively. And for Arc-hard, there are 1,119, 299, and 1,172 examples in the training, dev and testing sets respectively.

Table 5 presents the statistics of the above datasets, including data splitting size, the average input length (our unified QG encoding’s input length), and the average length of output (questions).

A. IMPLEMENTATION DETAILS

A.1 Dataset Details

Table 5 presents the statistics of the above datasets, including data splitting size, the average input length (our unified QG encoding’s input length), and the average length of output (questions).

SQuAD is one of the most commonly used answer-extraction QG datasets. There are two versions, and we use SQuADv1.1 [58] in our experiment, which contains 536 Wikipedia articles with more than 100k questions. Following [81], we split the dataset into training, dev, and testing sets, each set containing 87, 599, 5,280, and 5,285 elements. NarrativeQA [35] is a collection of about 1,500 stories and movie scripts with summaries. About 47,000 question and answer pairs are created by crowd-source workers. The dataset is split into training, dev, and testing sets with the size of 65,494, 6,922, and 21,114, respectively.
In this paper, we split the dataset into training, dev, and testing sets with the size of 9, 427, 1, 635, and 1, 635, respectively.

A.2 Model Implementation
All of our approaches are implemented based on PyTorch\(^2\). The backbone of our QG model is T5. Since T5 has a wide range of sizes version, we choose T5-base as the initial point of our QG model.

We utilize AdamW [44] as our optimizer with 3e − 4 learning rate. Gradient clipping is applied during the training period. For each dataset, we train at most 20 epochs, and if the validation loss does not decrease after 3 epochs, the training process will be early stopped. The batch size is 16. The max length of the input is set to 512. The original \(\lambda_{ori}\) for EWC term is set to 120000 in practice.

### B IMPROVING QA SYSTEMS USING UNIFIED-QG

One of the important applications of QG is to improve QA system’s performance via training data augmentation. Traditional QG models can only produce synthetic QA pairs for specific QA tasks. Therefore, in order to improve different kinds of QA systems, we have to construct, train, and store a large amount of QA models. As a result, the computational costs and storage space of the QG model are equal to or even greater than QA models'. In this part, we investigate how to use our single Unified-QG to improve 8 kinds of QA systems' performance, showing the efficiency of our Unified-QG.

To be simplified and without loss of generality, we use T5 as the backbone of our QG model and construct a unified QA encoding to let the model has the ability to address different QA tasks, instead of using dedicated QA models. Similar to our unified QG encoding, the input for QA model is also a string concatenation: “Question: ... + Passage: ... + [Distractor: ... +[...]]”, and the output target is the answer. Then, we train this QA model on different QA datasets respectively to investigate the benefits of QG augmentation.

To generate synthetic QA pairs, we follow the “back translation” approach in [65, 83]. We utilize our Unified-QG model to generate questions based on the context in original dataset, without introducing new articles. The low quality synthetic data are filtered out by using BLEU-4 scores. For all of the QA datasets, the size of the augmented dataset is two times of original one. Finally, we train QA model on the augmented dataset.

Following Ding et al’s work [18], we evaluate each dataset with the metrics most frequently used in previous works. Specifically, we employ exact match (EM) and ROUGE-L for extraction and abstraction QA. The boolean QA is evaluated by the accuracy of the “yes” or “no” label. The measure of multi-choice QA is relatively complex, since T5 directly generates the choice. We at first use word overlap count to find the closest choice for the generated answer, and then, we calculate how often it is correct.

Table 6 presents the results of improving 8 QA tasks via using Unified-QG generated data. From Table 6, we can observe that all the QA systems are improved after trained on combined datasets. To be specific, QA gains 0.35 EM scores on SQuAD, 0.40 ROUGE-L scores on NarrativeQA, 0.57 accuracy scores on RACE, 5.01 accuracy scores on McTest, 1.80 scores on OpenbookQA, 0.13 accuracy on Arc-easy, 0.09 scores on Arc-hard, and 1.96 accuracy on BoolQA. It is worth mentioning that since the aim of this experiment is to demonstrate that our single Unified-QG can simultaneously improve multiple QA tasks, which is infeasible for traditional QG models, therefore, the data filter method and synthetic data usages here are relatively simple. The advanced usage of QG for QA can be further studied in the future research.

---

\(^2\)https://pytorch.org/