Synthetic Question Value Estimation for Domain Adaptation of Question Answering

Anonymous ACL submission

Abstract

Synthesizing QA pairs with a question generator (QG) on the target domain has become a popular approach for domain adaptation of question answering (QA) models. Since synthetic questions are often noisy in practice, existing work adapts scores from a pretrained QA (or QG) model as criteria to select high-quality questions. However, these scores do not directly serve the ultimate goal of improving QA performance on the target domain. In this paper, we introduce a novel idea of training a *question value estimator (QVE)* that directly estimates the usefulness of synthetic questions for improving the target-domain QA performance. By conducting comprehensive experiments, we show that the synthetic questions selected by QVE can help achieve better target-domain QA performance, in comparison with existing techniques. We additionally show that by using such questions and only around 15% of the human annotations on the target domain, we can achieve comparable performance to the fully-supervised baselines.[1]

1 Introduction

Question answering (QA) systems based on pretrained language models such as BERT (Devlin et al., 2019) have recently achieved promising performance in machine reading comprehension. However, neural QA systems trained on one domain may not generalize well to another, leaving it challenging to deploy such systems on new domains that lack large-scale QA training data[2]. In this paper, we are interested in *semi-supervised domain adaptation*: we aim to build a target QA model with source-domain data and a small number of target-domain annotated QA pairs.

Due to high annotation costs, existing work (Golub et al., 2017; Dong et al., 2019; Wang et al., 2019; Puri et al., 2020; Chen et al., 2020; Yue et al., 2021) proposes to synthesize target-domain QA pairs via neural question generation (QG) models. The synthetic data are then used to train a QA model on the target domain. In practice, however, the generated questions are often of low quality, such as being semantically mismatched with their paired answers or asking about simple facts (Figure 1). Including all such questions for QA training is less likely to bring substantial improvements. This inspires us to study a crucial problem:

Given a set of target-domain synthetic QA pairs, how to select high-quality ones that are useful to improve target-domain QA training?

To address the problem, Alberti et al. (2019) propose the Roundtrip Consistency (RTC) method, which filters[3] questions that cannot be correctly answered by a pretrained QA model. Other work (Shakeri et al., 2020) considers using the generation log likelihood by the QG model (LM Score) as

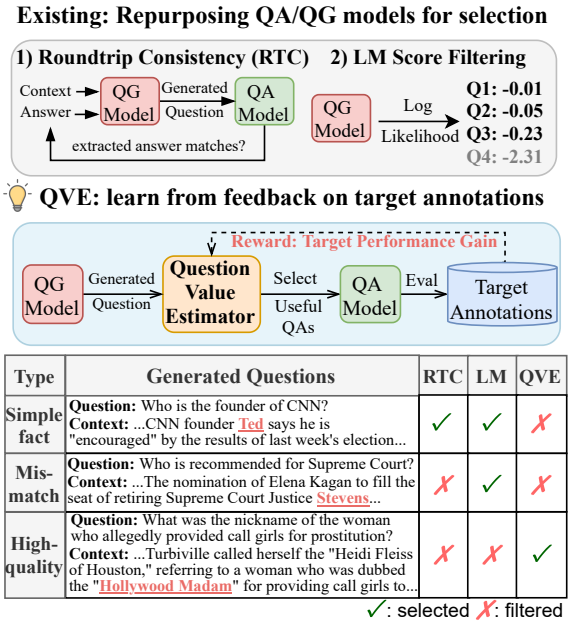

Figure 1: Existing work repurposes a pretrained QA (or QG) model to evaluate the quality of the generated questions, which is not directly associated with the target-domain QA performance and may select questions that are semantically-mismatched or ask about a simple fact. In contrast, our Question Value Estimator (QVE) learns to select useful questions with target-domain QA performance gain as direct feedback.

[1] Our source code will be released upon acceptance.

[2] Large-scale training data are typically 60-100K in size.

[3] We interchangeably use "filter" (noisy/low-quality questions) and "select" (useful/high-quality questions).

a metric to filter noisy questions (Figure 1, top). Although these filtering techniques have been shown to improve the question quality to some extent (Rennie et al., 2020), they are not directly optimized for *selecting questions that can improve QA performance on the target domain*. For example, some useful but difficult questions (e.g., the last example in Figure 1) may be filtered by the Roundtrip method, since they cannot be answered correctly by the pretrained QA model. However, these questions are often crucial to further improving QA performance when added into training.

In this paper, we propose a *question value estimator (QVE)* (Figure 1, middle) to select questions that can improve QA performance on the target domain. QVE takes in generated QA examples and outputs real-valued scores (i.e., question values), which are expected to represent the usefulness of generated questions in terms of improving target-domain QA performance. However, training the QVE model towards this goal is challenging due to the lack of supervision (i.e., true question values).

To solve the problem, we propose to train the QVE with direct QA feedback from the target domain. Intuitively, if a batch of synthetic questions (when used for training) leads to increasing accuracy of the target-domain QA model, QVE should assign high values to them; the more the accuracy increases, the higher the question values should be. Thus, we optimize QVE with the *target-domain QA performance gain* after adding the selected questions into training. More formally, given the discrete and non-differentiable question selection process, we formulate the question selection of QVE as a reinforcement learning (Williams, 1992) problem (Figure 2). The QVE receives a batch of synthetic samples each time and learns to select high-quality ones based on their estimated values. The selected samples are then used to train the target-domain QA model, with the resulting performance gain (on the available target-domain annotations) as the reward. The reward guides the optimization of QVE such that it will eventually make proper question value estimation and selection.

To evaluate the QVE model, we instantiate the QG and the QA model based on the pretrained BART (Lewis et al., 2020) and BERT (Devlin et al., 2019), respectively. By carrying out comprehensive experiments on four commonly-used reading comprehension datasets (Trischler et al., 2017; Joshi et al., 2017; Yang et al., 2018; Kwiatkowski et al., 2019), we show that: (1) our QVE model trained with the target-domain QA feedback substantially outperforms the question selection techniques trained without direct QA feedback (Alberti et al., 2019; Shakeri et al., 2020). (2) When using our QVE model to select synthetic questions, QA models can achieve comparable performance to fully-supervised baselines while using only 15% of the full target-domain annotations, which indicates that our method can greatly alleviate human annotation effort in practice. (3) To understand why QVE brings superior improvement, we conduct human evaluation and find that QVE can better identify semantically-matched and difficult questions.

2 Related Work

Domain Adaptation of Question Answering. In this field, some work (Wiese et al., 2017; Chung et al., 2018; Hazen et al., 2019; Cao et al., 2020) assumes that target-domain annotated questions are available, however, manually creating questions is costly. Therefore, another line of research work (Golub et al., 2017; Wang et al., 2019; Lee et al., 2020; Shakeri et al., 2020) investigates a domain adaptation setting where annotated questions are not available on the target domain. A commonly-adopted approach of this line is to leverage a neural question generation (QG) model (Du et al., 2017; Zhou et al., 2017; Sun et al., 2018; Zhao et al., 2018; Nema et al., 2019; Tuan et al., 2020) to automatically synthesize questions given unlabeled contexts (Du and Cardie, 2018; Zhang and Bansal, 2019; Wang et al., 2019; Liu et al., 2020; Golub et al., 2017; Wang et al., 2019; Lee et al., 2020; Shakeri et al., 2020; Yue et al., 2021); see more discussions in Section 3. However, it is very challenging to achieve satisfying performance without any target annotations. In our work, we study *semi-supervised domain adaptation of QA*, and assume *a small number of target annotations are available*, which can greatly help models adapt to the target domain while requiring minimal human effort.

Unsupervised and Semi-supervised QA are two other research topics relevant to our work (Fabbri et al., 2020; Li et al., 2020; Lewis et al., 2019; Dhingra et al., 2018). Unlike domain adaptation, these two settings do not assume the existence of the "source domain" and synthesize cloze-style questions via rule-based methods for building QA models. Since rule-based QG methods typically have much worse performance than neural ones

(pretrained on the source data), we do not compare with these two lines of research in experiments.

3 Background

3.1 Domain Adaptation of QA via QG

Semi-supervised Domain Adaptation. We study the semi-supervised domain adaptation of *extractive* question answering, where the source-domain and a small number[4] of target-domain QA annotations are provided. Formally, we denote the source-domain QA dataset as $D^{\mathsf{s}} = \{(c_i^{\mathsf{s}}, q_i^{\mathsf{s}}, a_i^{\mathsf{s}})\}_{i=1}^{N}$, where large-scale tuples of context c_i^{s}, question q_i^{s}, and answer a_i^{s} are available. For the target domain, only a small set of annotated QA pairs $D^{\mathsf{t}} = \{(c_j^{\mathsf{t}}, q_j^{\mathsf{t}}, a_j^{\mathsf{t}})\}_{j=1}^{M}$ are available ($M \ll N$). Since unlabeled contexts are easy to collect, we assume that they are largely available: $C^{\mathsf{t}} = \{c_l^{\mathsf{t}}\}_{l=1}^{L}$ ($L \gg M$). The task is to build a QA model that can accurately answer questions on the target domain, given D^{s}, D^{t}, and C^{t}.

Domain Adaptation via Question Generation. Given the lack of large-scale target-domain annotations, an intuitive approach to domain adaptation is first synthesizing target-domain QA data $D_{syn}^{\mathsf{t}} = \{(c_l^{\mathsf{t}}, q_l^{\mathsf{t}}, a_l^{\mathsf{t}})\}_{l=1}^{L}$ automatically from the unlabeled contexts C^{t}, and then training a target-domain QA model on the synthetic (D_{syn}^{t}) and the small-size annotated (D^{t}) target-domain data. In such an approach, a question generator (QG) g_ϕ is first pretrained on the source training data and further finetuned on the available target-domain annotated QA pairs. A well-trained QG model then takes target-domain context-answer pairs as input to generate a question: $q_l^{\mathsf{t}} = g_\phi(c_l^{\mathsf{t}}, a_l^{\mathsf{t}})$.

Although this approach has been shown promising, in practice, its effectiveness is restricted by the quality of synthetic questions. Thus, learning to select ones that can lead to a better target-domain QA model becomes a crucial problem.

With respect to how to obtain a_l^{t} for QG, in this paper, we assume an answer a_l^{t} (i.e., a text span in the context c_l^{t}) is given, following Du et al. (2017). When the answer a_l^{t} is not given, it can be extracted from the given context by using an entity recognition tool (Du and Cardie, 2018), a classifier (Puri et al., 2020) or a seq2seq model (Shakeri et al., 2020). Note that noise caused by such answer extraction tools will further lower the overall quality of the synthesized questions. In this paper, we focus on how to select useful synthetic questions in general (i.e., those questions can be synthesized by any QG process) and assume answers are given for simplicity.

[4] In our experiments, we assume 1,000 target annotations available, which is around 1-1.5% of the original training data.

3.2 Synthetic Question Selection

Given the synthetic target-domain QA data D_{syn}^{t}, the task is to select high-quality pairs from D_{syn}^{t} that are useful to improve target-domain QA training. Such a selection decision is often made based on some scores that can indicate the quality of the pairs. For example, Roundtrip filtering (Alberti et al., 2019) selects questions based on the extracted answer's correctness by a pretrained QA model. Similarly, LM filtering (Shakeri et al., 2020) selects questions with high log-likelihood scores in the generation. However, these scores do not directly serve the goal of improving target-domain QA training. Inspired by recent research on data selection in the machine learning community (Ghorbani and Zou, 2019; Jia et al., 2019; Yoon et al., 2020), we propose a new idea of training a *question value estimator*, which predicts the usefulness of a synthetic question for target-domain QA.

4 Question Value Estimator (QVE)

Formally, we design a question value estimator (QVE), e_γ, which takes in a synthetic QA example (c_l, q_l, a_l) (for simplicity, we omit the superscript t) and outputs a score indicating its "value," i.e., $v_l = e_\gamma(c_l, q_l, a_l)$. The "value" can imply "the potential for improving the target-domain QA performance when being used as a training sample". With this score, one can select most useful synthetic examples for the target-domain QA training.

We use a BERT model as the backbone of the QVE. Specifically, we concatenate the context, question and answer as input to the QVE, and use BERT to encode the sequence (Devlin et al., 2019).

$$\mathbf{h} = \texttt{BERT}\left[\texttt{<CLS>}\ q\ \texttt{<ANS>}\ a\ \texttt{<SEP>}\ c\right]$$

where q, a, c represent the question, answer, and context, respectively. $\mathbf{h} \in \mathbb{R}^H$ denotes the hidden representation of the input sequence derived from the "<CLS>" token. <ANS> and <SEP> are two special tokens used as delimiters.

In our preliminary experiments, we find that adding the answer (start index and end index) probabilities (p_s, p_e) by a pretrained QA model as additional features to the hidden representation $\mathbf{h}$ can

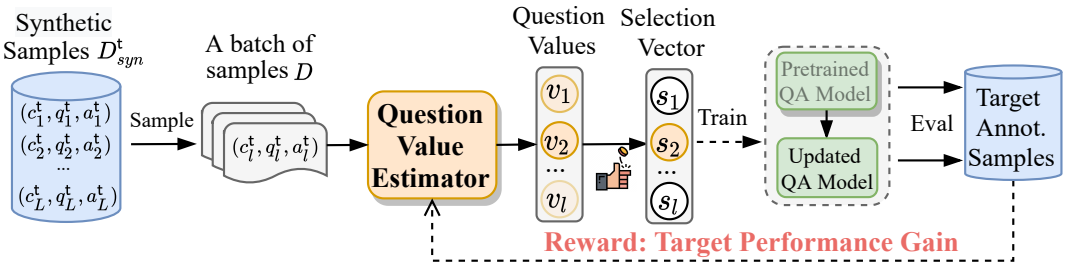

Figure 2: Illustration of QVE training based on the direct feedback from QA. Specifically, in the forward pass, QVE estimates the question values of a batch of synthetic questions and draws a Bernoulli sampling to select questions. The selected questions are then used to finetune a pretrained QA model. The performance gain (before and after the QA finetuning) on the target annotations is calculated as the reward for REINFORCED QVE training.

accelerate the QVE training convergence and lead to better performance. Thus, we add these two features (p_s, p_e) followed by linear transformations of the original hidden representation, and then build a linear classifier to output the question value.

$$\mathbf{h}' = \sigma(W_2\sigma(W_1\mathbf{h} + b_1) + b_2)$$
$$\mathbf{h}'' = \sigma(W_3(\mathbf{h}' \oplus p_s \oplus p_e) + b_3)$$
$$v_l = W_4\mathbf{h}'' + b_4$$

where $W_1 \in \mathbb{R}^{H_1 \times H}, W_2 \in \mathbb{R}^{H_2 \times H_1}, W_3 \in \mathbb{R}^{H_3 \times H_2}, W_4 \in \mathbb{R}^{H_3}$, $b_1 \in \mathbb{R}^{H_1}, b_2 \in \mathbb{R}^{H_2}, b_3 \in \mathbb{R}^{H_3}, b_4 \in \mathbb{R}$ are trainable parameters of linear layers. σ is the activation function `tanh`.

Learning such a question value estimator is challenging because we do not have direct supervision on the true value or usefulness of a synthetic question. We discuss two straightforward baselines to train QVE in Section 4.1, and a more advanced one based on reinforcement learning in Section 4.2.

4.1 QVE Training: Two Baselines

Binary Classifier: One straightforward solution is to treat QVE as a binary classifier and train it based on the human-annotated (positive) and the machine-synthesized (negative) QA pairs. Given the scarcity of target-domain data, we first pretrain the classifier on the source domain and then finetune it on the target domain. More specifically, we train a QG model on 70% of the source training data and generate synthetic questions on the remaining 30% of the source training contexts. The generated questions and the source-domain annotated questions are used to train this binary classifier. The classifier is then finetuned based on the small set of target-domain annotations (positive) and the samples synthesized on the same target-domain contexts (negative).

However, not all of the generated questions are bad. Simply treating all synthetic samples as negatives may mislead the classifier. Thus, we loose this assumption and introduce a ranking baseline.

Ranking Baseline: We assume that the quality of human-annotated questions is not inferior than that of machine-synthesized ones. Thus, we train QVE based on a ranking triplet loss defined as follows:

$$L_r = \sum \max(0, m + v_s - v_h)$$

where v_s, v_h are the estimated question values of the machine-synthesized sample and human-annotated sample. m is set to 0.15 as the margin.

The two baseline methods have two obvious drawbacks: (1) they are trained to differentiate between human-annotated and machine-synthesized samples, which is mismatched with our goal of selecting high-quality samples *among machine-synthesized data*; (2) similar as (Alberti et al., 2019; Shakeri et al., 2020), the two baselines are *not* trained with direct signals that can represent the usefulness of a synthetic question. In the next section, we will introduce a task-specific training method, which directly uses the target-domain QA feedback to optimize QVE.

4.2 QVE Training: Direct Feedback from QA

A well-trained QVE is expected to assign high values to synthetic questions that can improve the target-domain QA performance. Therefore, an intuitive way to measure the value of a synthetic question is to consider the downstream QA performance gain (on the available target annotations) before and after this question is included in the training set. However, this "leave-one-out" formulation is computationally expensive and time-consuming, given that it can estimate the value of only one single synthetic question in each forward pass. In light of this challenge, we instead estimate question values in a *batch-wise* fashion. Algorithm 1 and Figure 2 describe the learning process.

Algorithm 1 QVE REINFORCED Training

Input: pretrained QA model f_θ; target synthetic QA pairs D^{t}_{syn}; small target annotations D^{t}.
Hyperparameters: outer iterations I_o, outer batch size B_o, inner iterations I_n, inner batch size B_n, QVE learning rate α_o, QA learning rate α_n.
Output: QVE e_γ.

1: Randomly initialize e_γ
2: Store $\theta_0 \leftarrow \theta$ (pretrained QA checkpoint)
3: **for** outer iteration $= 1$ to I_o **do**
4: ▷ ① *Sample a batch of synthetic QA pairs:*
5: Sample $\mathcal{D} = \{(c_l, q_l, a_l)\}_{l=1}^{B_o}$ from D^{t}_{syn}
6: ▷ ② *Estimate question values:*
7: $\mathcal{V} = e_\gamma(\mathcal{D})$
8: ▷ ③ *Sample selection vector:*
9: $\mathcal{S} \sim \texttt{Bernoulli}(\mathcal{V})$
10: ▷ ④ *Update QA on selected samples:*
11: **for** inner iteration $= 1$ to I_n **do**
12: Sample $\{(c_l, q_l, a_l)\}_{l=1}^{B_n} \sim \mathcal{D}$
13: $\theta \leftarrow \theta - \frac{\alpha_n}{B_n} \sum_{l=1}^{B_n} s_l \cdot \nabla_\theta \mathcal{L}_{qa}$
14: **end for**
15: ▷ ⑤ *Calculate QA gain as QVE reward:*
16: $r_{qve} = \texttt{reward\_fn}(f_{\theta_0}, f_\theta, D^{\mathsf{t}})$
17: ▷ ⑥ *Update QVE based on Eq. 1:*
18: $\gamma \leftarrow \gamma - \alpha_o \cdot \nabla_\gamma \mathcal{L}_\gamma$
19: Reset $\theta \leftarrow \theta_0$
20: **end for**
21: **return** e_γ

Generally speaking, we frame the QVE model learning as a reinforcement learning problem (Williams, 1992), and stimulate QVE to assign higher values to more useful questions by using performance-driven rewards. Specially, for a batch of synthetic examples $\mathcal{D} = \{(c_l, q_l, a_l)\}_{l=1}^{B_o}$ in the outer training iteration (Line 4-5), the QVE model *selects a subset of examples* that are most likely to boost the QA performance on the target domain, based on its judgment on their values.

Mathematically, the decision-making outcome is represented by the selection vector $\mathcal{S} = (s_1, s_2, ..., s_{B_o})$, where $s_l \in \{0, 1\}$ $l = 1, ..., B_o$ (Line 6-9). The whole batch-level decision making policy π_γ is described as follows:

$$v_l = e_\gamma(c_l, q_l, a_l)$$
$$s_l \sim \texttt{Bernoulli}(v_l)$$
$$\pi_\gamma(\mathcal{S}|\mathcal{D}) = \prod_{l=1}^{B_o} [v_l^{s_l} \cdot (1 - v_l)^{1-s_l}],$$

where the selection of a certain example (c_l, q_l, a_l) is formulated as sampling from a Bernoulli distribution of probability v_l (i.e., its estimated question value). We adopt the Bernoulli sampling based on the estimated value v_l instead of setting a hard threshold to encourage the policy exploration.

The model is rewarded based on how much performance gain the selected examples could bring when they are used to train the target-domain QA model. To this end, we finetune the QA model f_θ on the selected batch samples based on $\mathcal{L}_{qa}$, which typically is a cross-entropy loss:

$$\mathcal{L}_{qa} = -\sum_{l}^{B_o} \log P(a_l|q_l, c_l; \theta)$$

In practice, to stabilize the QVE training, we choose a large outer batch size B_o in each outer training iteration. For finetuning the QA model, we pick a relatively smaller inner batch size B_n and repeat the training for I_n times, such that the QVE-selected samples are fully utilized (Line 10-14).

The reward r_{qve} is defined as the QA performance gain on the target-domain annotations D^{t} before (f_{θ_0}) and after (f_θ) finetuning (Line 15-16),

$$r_{qve} = \texttt{reward\_fn}(f_{\theta_0}, f_\theta, D^{\mathsf{t}})$$

where `reward_fn` is Exact Match (EM) gain[5].

Given the discrete and non-differentiable question selection process, we update the QVE model using the REINFORCE algorithm (Williams, 1992). Mathematically, we aim to minimize:

$$\mathcal{L}_\gamma = -\mathop{\mathbb{E}}_{\mathcal{S}\sim\pi_\gamma(\cdot|\mathcal{D})}[r_{qve}].$$

The gradient of the loss function is derived as:

$$\begin{aligned}\nabla_\gamma \mathcal{L}_\gamma &= -\mathop{\mathbb{E}}_{\mathcal{S}\sim\pi_\gamma}[r_{qve} \nabla_\gamma \log \pi_\gamma(\mathcal{S}|\mathcal{D})] \\ &= -\mathop{\mathbb{E}}_{\mathcal{S}\sim\pi_\gamma}[r_{qve} \nabla_\gamma \sum_{l=1}^{B_o} \log[v_l^{s_l}(1 - v_l)^{1-s_l}]].\end{aligned} \quad (1)$$

Notably, to mitigate the instability in reinforcement learning, we reset the QA model to its pretrained checkpoint at the end of each outer iteration (Line 19), and keep the pretrained QG model unchanged.

After training QVE, we can use it to calculate the question value for all the synthetic questions on the target domain. Then we can select top $K\%$ synthetic QA pairs as the training corpus to train the target-domain QA model.

[5]We also tried F1 gain and loss drop as the `reward_fn` and the EM gain is slightly better than the other two.

5 Experimental Setup

5.1 Datasets

We use datasets in the MRQA 2019 Shared Task (Fisch et al., 2019), a popular challenge focusing on generalization in reading comprehension. Specifically, following Shakeri et al. (2020), we use **SQuAD 1.1** (Rajpurkar et al., 2016) as the *source-domain* dataset. For the *target-domain* datasets, we consider **NewsQA** (Trischler et al., 2017), **Natural Questions (NQ)** (Kwiatkowski et al., 2019), **HotpotQA** (Yang et al., 2018) and **TriviaQA** (Joshi et al., 2017) as they are commonly used and have sufficient contexts for the QG model to generate synthetic samples. Since there is no test set available for each dataset, we use the original dev set as the test set. Detailed descriptions of each dataset are in Appendix A.

For the target-domain datasets, we assume all the contexts and n annotated QA pairs in the original training sets are available for training. We set $n = 1000$ (about 1%-1.5% of original training sets) as default and discuss the impact of n in Section 6.2.

5.2 Implementation Details

We implement models using the Hugging Face transformer (Wolf et al., 2020) library. We instantiate the QA model with `BERT-base-uncased` (Devlin et al., 2019), and the QG model with `BART-base` (Lewis et al., 2020). For QVE, we use a 4-layer transformer model instead of `BERT-base` since: (1) a smaller model allows a larger outer batch size B_o in training (given the GPU memory constraint[6]), which can also lead to more steady and efficient training; (2) we do not observe significant improvement when using `BERT-base` model in our preliminary study. We set $H_1 = H_3 = H = 512$ and $H_2 = 64$ for linear layers of QVE. For QVE training (Algorithm 1), we set $I_o = 2000$, $B_o = 120$, $I_n = 20$, $B_n = 12$, and $\alpha_o = \alpha_n = 3e^{-5}$. When training (finetuning) QA and QG models (either on source or target domain), we set training epochs as 3 and other hyperparameters as default in the transformer library.

5.3 Comparing Baselines

We evaluate the following QA models built on different training data:

(1) Source Only Baseline: we train a QA model on the source-domain data.

[6]We train all models on 4 GTX 1080 Ti 11GB GPUs.

	Different Filtering Methods			
Dataset	NoFilter	RTC	LM	QVE
NewsQA	74,160	33,756	44,485	44,485
NQ	104,071	62,888	62,443	62,443
HotpotQA	72,928	46,273	43,757	43,757
TriviaQA	61,688	26,361	37,013	37,013

Table 1: Number of synthetic examples selected by different methods. NoFilter: QG baseline (no filtering); RTC: Roundtrip Filtering; LM: LM Filtering.

(2) Source + Target Annotations Baseline: we further finetune the "(1) Source Only Baseline" on the available target annotated QA pairs.

(3) QG Baseline (no filtering): we first pretrain a QG model on the source-domain data and finetune it on the available target annotations. The QG model is then used to generate synthetic QA samples on the target contexts. We finetune a QA model sequentially on all available data with the order of "source→target synthetic→target annotated"[7]. The same QA finetuning strategy will also be used for (4)-(8).

(4) RoundTrip Filtering (Alberti et al., 2019): we use the "(2) Source + Target Annotation Baseline" to extract answers for target synthetic questions and select the ones, whose extracted answers are correct, as the target synthetic training corpus.

(5) LM Filtering (Shakeri et al., 2020): we use the log likelihood scores of synthetic questions produced by the QG model in (3) as the filtering criterion. We select top K% samples as the target synthetic training corpus.

(6) QVE (binary classifier): we train QVE as a binary classifier (Section 4.1) and then use it to select top K% target synthetic samples.

(7) QVE (ranking baseline): we train QVE based on a ranking function (Section 4.1), and then use it to select top K% synthetic samples.

(8) QVE (RL): we train QVE based on the direct feedback from target annotations using RL (Section 4.2), and then use it to select top K% target synthetic samples.

(9) Fully-supervised Baseline: we train a QA model on the original target training data. Note that we report the fully-supervised performance here only as the reference and (1)-(8) are not directly comparable to this.

The number of the selected synthetic examples of RoundTrip Filtering is determined by the QA model and varies for each dataset. For LM Filter-

[7]We also try combining all the data into one training file to finetune the QA model but the performance is lower than the current strategy.

No.	Methods	NewsQA		NQ		HotpotQA		TriviaQA	
		EM	F1	EM	F1	EM	F1	EM	F1
(1)	Source Only Baseline	40.2	56.2	45.2	59.1	43.3	60.3	49.5	59.3
(2)	Source + Target Annotations Baseline	43.7	59.8	54.2	68.2	51.7	69.2	55.7	62.0
(3)	QG Baseline (no filtering)	45.3	60.7	60.5	72.6	52.9	70.0	58.3	63.9
(4)	+RoundTrip Filtering (Alberti et al., 2019)	45.4	60.8	58.6	71.2	53.9	70.5	58.7	64.4
(5)	+LM Filtering (Shakeri et al., 2020)	45.3	61.2	60.0	72.1	53.9	70.5	56.0	61.7
(6)	+QVE (binary classifier)	45.2	60.7	60.1	72.3	53.7	70.4	58.2	63.8
(7)	+QVE (ranking baseline)	45.8	61.3	60.6	72.8	53.9	70.9	58.4	63.9
(8)	**+QVE (RL)**	**46.9**	**61.9**	**61.3**	**73.2**	**54.9**	**71.8**	**61.3**	**66.9**
(9)	Fully-supervised Baseline	50.0	64.6	65.8	78.1	56.8	73.9	64.6	70.3

Table 2: Semi-supervised domain adaptation performance of different models where 1,000 target-domain annotations (around 1-1.5% of the original training data) are used.

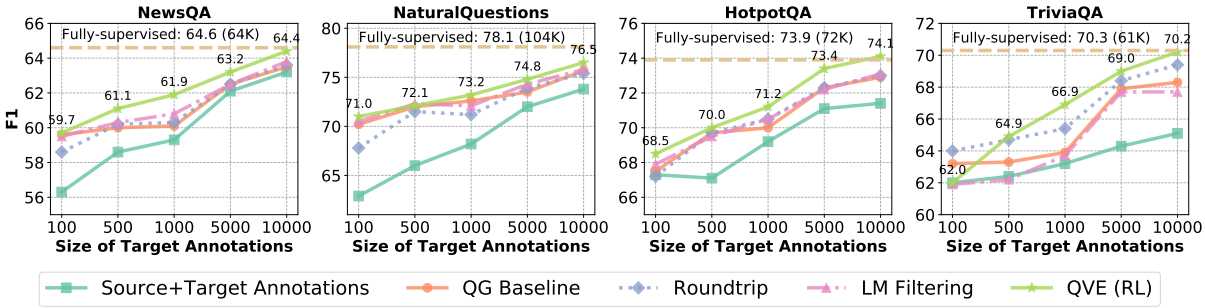

Figure 3: Impact of the number of target annotated QA pairs. We also show the fully-supervised performance (and #train) as the reference. With 10K target annotations (around 15% of the full training set), our method can achieve comparable performance to the supervised ones (as shown at the top of each sub-figure).

ing and QVE, we select top K% (K=60) samples among all synthetic ones and discuss the impact of the synthetic dataset size in Appendix B. We show the statistics of filtered datasets in Table 1.

6 Results

6.1 Overall Results

We first discuss the domain adaptation results on the 4 target-domain QA datasets under semi-supervised setting where $n = 1,000$ target-domain QA examples are available. Table 2 shows the overall results of different methods. We summarize key findings as follows:

(1) Compared with RoundTrip and LM Filtering, our QVE (RL) achieves the best performance. This is because both baselines are not specifically trained to select useful examples for improving QA performance on the target domain. Our QVE, on the contrary, is trained with a signal that directly reflects the QA performance, which can more accurately estimate the question value and select useful pairs for target-domain QA.

(2) Two QVE baselines (binary classifier and ranking baseline) can select some useful questions and achieve comparable performance with RoundTrip and LM Filtering. However, due to the lack of direct QA evaluation feedback, they underperform QVE (RL), which demonstrates the usefulness of the QA feedback during training QVE.

6.2 How many target QA pairs do we need?

In Table 2, we showed that with n (n=1,000) target annotated QA pairs and the selected high-quality synthetic QA pairs, we can finetune a better QA model on the target domain. In this section, we discuss the influence of n on the target-domain QA performance. The results are shown in Figure 3, and interesting findings include:

(1) In general, the performance of all models improves as more target annotations are used. This is intuitive as more annotated pairs can improve both QA and QG training. With a better QG model, the quality of the synthetic questions is improved, which could also lead to better QA models.

(2) Our QVE model can often outperform the QG baseline and the filtering baselines. With an optimization objective considering the downstream QA performance, QVE can select more useful questions for improving target-domain QA.

(3) The improvement of our QVE compared with baselines is usually larger when more annotated QA pairs are available. This is because our QVE training (with RL) relies on the QA feedback based

Setups	Methods	NQ		HotpotQA	
		EM	F1	EM	F1
QA:Large Model QG:Base Model	Source Only	50.7	65.0	46.2	64.0
	+ Target Annot.	58.7	72.1	54.3	72.2
	+ QG Baseline	61.6	73.4	55.5	72.5
	+ Roundtrip	59.8	71.9	55.9	72.8
	+ LM Filtering	60.6	72.5	55.7	72.7
	+ QVE (RL)	**62.4**	**74.5**	**56.3**	**73.4**
QA:Base Model QG:Large Model	Source Only	45.2	59.1	43.3	60.3
	+ Target Anno.	54.2	68.2	51.7	69.2
	+ QG Baseline	61.0	72.8	53.2	70.9
	+ Roundtrip	59.9	71.7	54.1	71.1
	+ LM Filtering	60.6	72.2	54.2	71.2
	+ QVE (RL)	**62.1**	**73.8**	**55.2**	**72.0**

Table 3: Results on larger capacity QG and QA models.

on the available annotated pairs. With more annotated pairs, the feedback can be more accurate, thus leading to a better QVE for selecting more useful synthetic questions.

(4) With 10,000 (around 15% of the original training set) target annotations and the synthetic questions selected by QVE, we can achieve comparable performance with the fully-supervised baseline. This indicates that one can save more annotation budgets when building a target-domain QA model based on our QVE in practice.

6.3 Experiments with Larger Models

The results presented in the previous sections are based on `BERT-base` and `BART-base`. In this section, we test whether our QVE can still be effective when working with larger models, and select `BERT-Large` and `BART-Large` as QA and QG model respectively. When changing the QA (QG) model to its larger alternative, we keep the other one as the base model to better show the difference. We use NaturalQuestions (NQ) and HotpotQA as representative datasets, and show results on them (with 1,000 target annotations). As shown in Table 3, our QVE model can still help improve the performance for larger instantiations of QG/QA.

6.4 Human Study: Why can QVE help QA?

In this section, we aim to gain a better understanding of why QVE helps QA and verify that QVE selects more semantically matched and non-trivial questions, thus benefiting downstream QA.

Since automatic metrics cannot often reflect the actual quality of the question selections, we sample 50 generated examples from each target-domain dataset (200 in total), and ask three human annotators to label whether a generated QA pair is semantically matched (i.e., can be selected to train QA) and (if yes) whether it asks about a simple fact. To lower the annotation bias in determining whether a generated question asks about a simple fact or not, we provide the ground-truth question (the question in the original dataset created by humans) as a reference. If the generated question is simpler than the ground truth, then it would be marked as "trivial"; otherwise, it is a "non-trivial" one. Three annotators work independently and we adopt the majority vote for deciding the final labels of a generated QA pair (if disagreement appears).

Methods	Semantically-Matched			Non-trivial		
	P	R	F1	P	R	F1
RoundTrip	**88.9**	60.6	72.1	82.6	47.5	60.3
LM Filtering	86.7	65.0	74.3	78.9	51.7	62.5
QVE(RL)	**88.9**	**70.0**	**78.3**	**83.3**	**59.3**	**69.3**

Table 4: Agreement with question selection by humans.

We calculate the precision, recall and F1 between predictions[8] by each filtering method and human labels (for both "semantically matched" and "non-trivial"). As shown in Table 4, though three methods obtain a similar precision on all sampled questions, our method has a better recall, especially on the "non-trivial" questions. This means that our method can select more semantically matched and non-trivial questions, which explains why it leads to better QA performance. We also show some real cases in Figure 1 to further illustrate this point. For example, our QVE selects *"What was the nickname given to the woman who allegedly provided call girls for prostitution?"* while the baselines do not pick this semantically matched and non-trivial question. For another example, *"Who is the founder of CNN"*, both baselines select it while our QVE filters it out since such a simple question would probably not help further improve QA.

7 Conclusion

We propose a question value estimator to estimate the usefulness of synthetic questions and select useful ones for improving target-domain QA training. We optimize QVE with the target-domain QA performance gain after adding the selected questions into training. Our comprehensive experiments demonstrate the superiority of QVE compared with other question selection methods. Additionally, using the synthetic questions selected by QVE and only around 15% of the human annotated data on each target domain, we can achieve comparable performance to the fully-supervised baselines.

[8] We treat it as a binary classification problem here: if a question is selected, the prediction is 1; 0 otherwise.

References

Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. 2019. Synthetic QA corpora generation with roundtrip consistency. In *ACL'19*, pages 6168–6173. Association for Computational Linguistics.

Yu Cao, Meng Fang, Baosheng Yu, and Joey Tianyi Zhou. 2020. Unsupervised domain adaptation on reading comprehension. In *AAAI'20*.

Yanda Chen, Md. Arafat Sultan, and Vittorio Castelli. 2020. Improved synthetic training for reading comprehension. *CoRR*, abs/2010.12776.

Yu-An Chung, Hung-Yi Lee, and James Glass. 2018. Supervised and unsupervised transfer learning for question answering. In *NAACL-HLT'18*, pages 1585–1594.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT'19*, pages 4171–4186. Association for Computational Linguistics.

Bhuwan Dhingra, Danish Danish, and Dheeraj Rajagopal. 2018. Simple and effective semi-supervised question answering. In *NAACL'18*, pages 582–587.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *NeurIPS'19*, pages 13042–13054.

Xinya Du and Claire Cardie. 2018. Harvesting paragraph-level question-answer pairs from wikipedia. In *ACL'18*, pages 1907–1917.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *ACL'17*, pages 1342–1352.

Alexander Richard Fabbri, Patrick Ng, Zhiguo Wang, Ramesh Nallapati, and Bing Xiang. 2020. Template-based question generation from retrieved sentences for improved unsupervised question answering. In *ACL'20*, pages 4508–4513.

Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. 2019. MRQA 2019 shared task: Evaluating generalization in reading comprehension. In *MRQA@EMNLP'19*, pages 1–13. Association for Computational Linguistics.

Amirata Ghorbani and James Y. Zou. 2019. Data shapley: Equitable valuation of data for machine learning. In *ICML'19*, volume 97 of *Proceedings of Machine Learning Research*, pages 2242–2251. PMLR.

David Golub, Po-Sen Huang, Xiaodong He, and Li Deng. 2017. Two-stage synthesis networks for transfer learning in machine comprehension. In *EMNLP'17*, pages 835–844. Association for Computational Linguistics.

Timothy J Hazen, Shehzaad Dhuliawala, and Daniel Boies. 2019. Towards domain adaptation from limited data for question answering using deep neural networks. *arXiv preprint arXiv:1911.02655*.

Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J. Spanos. 2019. Towards efficient data valuation based on the shapley value. In *AISTATS'19*, volume 89 of *Proceedings of Machine Learning Research*, pages 1167–1176. PMLR.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL'17*, pages 1601–1611. Association for Computational Linguistics.

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

Dong Bok Lee, Seanie Lee, Woo Tae Jeong, Donghwan Kim, and Sung Ju Hwang. 2020. Generating diverse and consistent QA pairs from contexts with information-maximizing hierarchical conditional vaes. In *ACL'20*, pages 208–224. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL'20*, pages 7871–7880. Association for Computational Linguistics.

Patrick S. H. Lewis, Ludovic Denoyer, and Sebastian Riedel. 2019. Unsupervised question answering by cloze translation. In *ACL'19*, pages 4896–4910. Association for Computational Linguistics.

Zhongli Li, Wenhui Wang, Li Dong, Furu Wei, and Ke Xu. 2020. Harvesting and refining question-answer pairs for unsupervised qa. In *ACL'20*, pages 6719–6728.

Bang Liu, Haojie Wei, Di Niu, Haolan Chen, and Yancheng He. 2020. Asking questions the human way: Scalable question-answer generation from text corpus. In *WWW'20*, pages 2032–2043.

Preksha Nema, Akash Kumar Mohankumar, Mitesh M Khapra, Balaji Vasan Srinivasan, and Balaraman Ravindran. 2019. Let's ask again: Refine network for automatic question generation. In *EMNLP-IJCNLP'19*, pages 3305–3314.

Raul Puri, Ryan Spring, Mohammad Shoeybi, Mostofa Patwary, and Bryan Catanzaro. 2020. Training question answering models from synthetic data. In *EMNLP'20*, pages 5811–5826. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP'16*, pages 2383–2392. The Association for Computational Linguistics.

Steven J. Rennie, Etienne Marcheret, Neil Mallinar, David Nahamoo, and Vaibhava Goel. 2020. Unsupervised adaptation of question answering systems via generative self-training. In *EMNLP'20*, pages 1148–1157. Association for Computational Linguistics.

Siamak Shakeri, Cícero Nogueira dos Santos, Henghui Zhu, Patrick Ng, Feng Nan, Zhiguo Wang, Ramesh Nallapati, and Bing Xiang. 2020. End-to-end synthetic data generation for domain adaptation of question answering systems. In *EMNLP'20*, pages 5445–5460. Association for Computational Linguistics.

Xingwu Sun, Jing Liu, Yajuan Lyu, Wei He, Yanjun Ma, and Shi Wang. 2018. Answer-focused and position-aware neural question generation. In *EMNLP'18*, pages 3930–3939. Association for Computational Linguistics.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. Newsqa: A machine comprehension dataset. In *Rep4NLP@ACL'17*, pages 191–200. Association for Computational Linguistics.

Luu Anh Tuan, Darsh J. Shah, and Regina Barzilay. 2020. Capturing greater context for question generation. In *AAAI'20*, pages 9065–9072. AAAI Press.

Huazheng Wang, Zhe Gan, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, and Hongning Wang. 2019. Adversarial domain adaptation for machine reading comprehension. In *EMNLP-IJCNLP'19*, pages 2510–2520. Association for Computational Linguistics.

Georg Wiese, Dirk Weissenborn, and Mariana Neves. 2017. Neural domain adaptation for biomedical question answering. In *CoNLL'17*, pages 281–289.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8:229–256.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *EMNLP'20*, pages 38–45, Online. Association for Computational Linguistics.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *EMNLP'18*, pages 2369–2380. Association for Computational Linguistics.

Jinsung Yoon, Sercan Arik, and Tomas Pfister. 2020. Data valuation using reinforcement learning. In *ICML'20*, pages 10842–10851. PMLR.

Zhenrui Yue, Bernhard Kratzwald, and Stefan Feuerriegel. 2021. Contrastive domain adaptation for question answering using limited text corpora. In *EMNLP'21*.

Shiyue Zhang and Mohit Bansal. 2019. Addressing semantic drift in question generation for semi-supervised question answering. In *EMNLP-IJCNLP'19*, pages 2495–2509. Association for Computational Linguistics.

Yao Zhao, Xiaochuan Ni, Yuanyuan Ding, and Qifa Ke. 2018. Paragraph-level neural question generation with maxout pointer and gated self-attention networks. In *EMNLP'18*, pages 3901–3910. Association for Computational Linguistics.

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

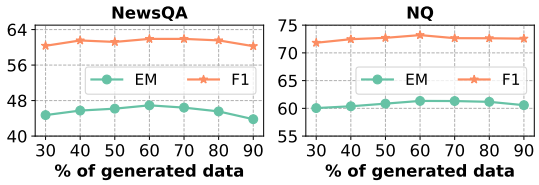

Figure A1: Impact of synthetic dataset size.

A Details of Datasets

Specifically, following Shakeri et al. (2020), we use **SQuAD 1.1** (Rajpurkar et al., 2016), a large reading comprehension dataset that consists of 100k questions on more than 500 articles from Wikipedia, as the *source-domain* dataset. For the *target-domain* datasets, we consider the following 4 datasets since they are commonly used and have sufficient contexts to train the models.

NewsQA (Trischler et al., 2017) consists of questions and answers based on a set of over 10k news articles from CNN News.

Natural Questions (NQ) (Kwiatkowski et al., 2019) contains questions extracted from Google user search queries and passages from Wikipedia.

HotpotQA (Yang et al., 2018) is a multi-hop question answering dataset based on Wikipedia passages.

TriviaQA (Joshi et al., 2017) includes QA pairs authored by trivia enthusiasts, as well as evidence documents independently gathered from Web search results and Wikipedia articles.

B Impact of Synthetic Dataset Size

In Figure A1, we show how the synthetic dataset size (i.e., the number of selected QA pairs) impacts the QA performance, based on our QVE (RL) filtering. As we expect, at the beginning, the target QA performance improves when more synthetic data is added to the training set. However, the performance reaches the peak at 60-70% and then goes down. This is reasonable since adding less valuable QA pairs from the noisy synthetic data will hurt the QA model training. We suggest 60%-70% (50K-70K QA pairs) for setting the synthetic data size in practical.