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 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. 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 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 a popular approach for domain adaptation of 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?

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.
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 eventually makes 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...
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 \(D^s = \{(c_i^s, q_i^s, a_i^s)\}_{i=1}^N\), 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 \(D^t = \{(c_j^t, q_j^t, a_j^t)\}_{j=1}^M\) are available (\(M \ll N\)). Since unlabeled contexts are easy to collect, we assume that they are largely available: \(C^t = \{c_i^t\}_{i=1}^L\) (\(L \gg M\)). 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 \(D_{syn}^t = \{(c_i^t, q_i^t, a_i^t)\}_{i=1}^M\) 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) \(g_{\phi}\) is first pretrained on the source training data and further fine-tuned 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: \(q_i^t = g_{\phi}(c_i^t, a_i^t)\).

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 \(a_i^s\) for QG, in this paper, we assume an answer \(a_i^s\) (i.e., a text span in the context \(c_i^s\)) is given, following Du et al. (2017). When the answer \(a_i^s\) 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.

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 (Ghobami 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), \(\psi\), which takes in a synthetic QA example \((c_t, q_t, a_t)\) (for simplicity, we omit the superscript \(t\)) and outputs a score indicating its “value,” i.e., \(v_t = \psi(c_t, q_t, a_t)\). 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).

\[
\text{h} = \text{BERT} \begin{bmatrix} <\text{CLS}> & q <\text{ANS}> & a <\text{SEP}> & c \end{bmatrix}
\]

where \(q, a, c\) represent the question, answer, and context, respectively. \(h \in \mathbb{R}^H\) 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 \(h\) can
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.

\[
h' = \sigma(W_2 \sigma(W_1 h + b_1) + b_2)
\]

\[
h'' = \sigma(W_3 (h' \odot p_s \odot p_e) + b_3)
\]

\[
v_t = W_4 h'' + b_4
\]

where \(W_1 \in \mathbb{R}^{H_1 \times H}, W_2 \in \mathbb{R}^{H_2 \times H_1}, W_3 \in \mathbb{R}^{H_2 \times H_2}, W_4 \in \mathbb{R}^{H_3}, b_1 \in \mathbb{R}^{H_1}, b_2 \in \mathbb{R}^{H_2}, b_3 \in \mathbb{R}^{H_3}, b_4 \in \mathbb{R}\) are trainable parameters of linear layers. \(\sigma\) 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 fine-tune 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 $f_\theta$; target synthetic QA pairs $D^*_\text{syn}$; small target annotations $D^t$.
Hyperparameters: outer iterations $I_o$, outer batch size $B_o$, inner iterations $I_i$, inner batch size $B_i$, QVE learning rate $\alpha_o$, QA learning rate $\alpha_i$.

Output: QVE $e_\gamma$.

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

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 $D = \{(c_l, q_l, a_l)\}_{l=1}^{B_o}$ 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 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 $f_\theta$ on the selected batch samples based on $L_{qa}$, which typically is a cross-entropy loss:

$$L_{qa} = -\sum_l \log P(a_l|q_l, c_l; \theta)$$

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 $B_i$ and repeat the training for $I_i$ 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 $(f_\theta)$ and after $(f_\theta)$ finetuning (Line 15-16),

$$r_{qve} = \text{reward}_f(f_\theta, f_\theta, D^t)$$

where $\text{reward}_f$ 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:

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

The gradient of the loss function is derived as:

$$\nabla_\gamma L_\gamma = -\mathbb{E}_{S \sim \pi_\gamma} \left[r_{qve}\nabla_\gamma \log \pi_\gamma(S|D)\right]$$

$$= -\mathbb{E}_{S \sim \pi_\gamma} \left[r_{qve}\nabla_\gamma \sum_l \log [v_l^n(1 - v_l)^{1-n}]\right].$$

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 $\text{reward}_f$ 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 comprehension reading. 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.

(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-

\(^6\)We train all models on 4 GTX 1080 Ti 11GB GPUs.

\(^7\)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.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>NoFilter</th>
<th>RTC</th>
<th>LM</th>
<th>QVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NewsQA</td>
<td>74,160</td>
<td>33,756</td>
<td>44,485</td>
<td>44,485</td>
</tr>
<tr>
<td>NQ</td>
<td>104,071</td>
<td>62,888</td>
<td>64,443</td>
<td>64,443</td>
</tr>
<tr>
<td>HotpotQA</td>
<td>72,928</td>
<td>46,273</td>
<td>43,757</td>
<td>43,757</td>
</tr>
<tr>
<td>TriviaQA</td>
<td>61,688</td>
<td>26,361</td>
<td>37,013</td>
<td>37,013</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>No.</th>
<th>Methods</th>
<th>NewsQA</th>
<th>NQ</th>
<th>HotpotQA</th>
<th>TriviaQA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>EM</td>
<td>F1</td>
<td>EM</td>
<td>F1</td>
</tr>
<tr>
<td>(1)</td>
<td>Source Only Baseline</td>
<td>40.2</td>
<td>56.2</td>
<td>45.2</td>
<td>59.1</td>
</tr>
<tr>
<td>(2)</td>
<td>Source + Target Annotations Baseline</td>
<td>43.7</td>
<td>59.8</td>
<td>54.2</td>
<td>68.2</td>
</tr>
<tr>
<td>(3)</td>
<td>QG Baseline (no filtering)</td>
<td>45.3</td>
<td>60.7</td>
<td>60.5</td>
<td>72.6</td>
</tr>
<tr>
<td>(4)</td>
<td>+RoundTrip Filtering (Alberti et al., 2019)</td>
<td>45.4</td>
<td>60.8</td>
<td>58.6</td>
<td>71.2</td>
</tr>
<tr>
<td>(5)</td>
<td>+LM Filtering (Shakeri et al., 2020)</td>
<td>45.3</td>
<td>61.2</td>
<td>60.0</td>
<td>72.1</td>
</tr>
<tr>
<td>(6)</td>
<td>+QVE (binary classifier)</td>
<td>45.2</td>
<td>60.7</td>
<td>60.1</td>
<td>72.3</td>
</tr>
<tr>
<td>(7)</td>
<td>+QVE (ranking baseline)</td>
<td>45.8</td>
<td>61.3</td>
<td>60.6</td>
<td>72.8</td>
</tr>
<tr>
<td>(8)</td>
<td>+QVE (RL)</td>
<td>46.9</td>
<td>61.9</td>
<td>61.3</td>
<td>73.2</td>
</tr>
<tr>
<td>(9)</td>
<td>Fully-supervised Baseline</td>
<td>50.0</td>
<td>64.6</td>
<td>65.8</td>
<td>78.1</td>
</tr>
</tbody>
</table>

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).

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...
We use NaturalQuestions (NQ) and HotpotQA as (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.

6.3 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 model based on our QVE in practice.

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).

We calculate the precision, recall and F1 between predictions 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 “Who is 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.

---

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

<table>
<thead>
<tr>
<th>Setups</th>
<th>Methods</th>
<th>NQ</th>
<th>HotpotQA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>EM</td>
<td>F1</td>
</tr>
<tr>
<td>QA:Large Model</td>
<td>Source Only</td>
<td>50.7</td>
<td>65.0</td>
</tr>
<tr>
<td></td>
<td>+ Target Anno.</td>
<td>56.7</td>
<td>72.1</td>
</tr>
<tr>
<td></td>
<td>+ QG Baseline</td>
<td>61.6</td>
<td>75.4</td>
</tr>
<tr>
<td></td>
<td>+ Roundtrip</td>
<td>59.8</td>
<td>71.9</td>
</tr>
<tr>
<td></td>
<td>+ LM Filtering</td>
<td>60.6</td>
<td>72.2</td>
</tr>
<tr>
<td></td>
<td>+ QVE (RL)</td>
<td>62.4</td>
<td>74.5</td>
</tr>
</tbody>
</table>

| QA:Base 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 4: Agreement with question selection by humans.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Semantically-Matched</th>
<th>Non-trivial</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td>RoundTrip</td>
<td>88.9</td>
<td>60.6</td>
</tr>
<tr>
<td>LM Filtering</td>
<td>86.7</td>
<td>65.0</td>
</tr>
<tr>
<td>QVE(RL)</td>
<td>88.9</td>
<td>70.0</td>
</tr>
</tbody>
</table>

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


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.