Select High-quality Synthetic QA Pairs to Augment Training Data in MRC Under the Reward Guidance of Generative Language Models

Jing Jin, Houfeng Wang
National Key Laboratory of Multimedia Information Processing
School of Computer Science, Peking University
{11jj617, wanghf}@pku.edu.cn

Abstract

Synthesizing QA pairs via question generator (QG) for data augmentation is widely used in Machine Reading Comprehension (MRC), especially in data-scarce scenarios like limited labeled data or domain adaptation. However, the quality of generated QA pairs varies, and it is necessary to select the ones with high quality from them. Existing approaches focus on downstream metrics to choose QA pairs, which lacks generalization across different metrics and datasets. In this paper, we propose a general selection method that employs a generative large pre-trained language model as a reward model in a Reinforcement Learning (RL) framework for the training of the selection agent. Our experiments on both generative and extractive datasets demonstrate that our selection method leads to better downstream performance. We also find that using the large language model (LLM) as a reward model is more beneficial than using it as a direct selector or QA model. Furthermore, we assess the selected QA pairs from multiple angles, not just downstream metrics, highlighting their superior quality compared to other methods. Our work has better flexibility across metrics, provides interpretability for the selected data, and expands the potential of leveraging generative large language models in the field of MRC and RL training. Our code is available at `https://github.com/JulieJin-km/LLM_RL_Selection`.

Keywords: Machine Reading Comprehension, Data Selection, Generative Large Language Model

1. Introduction

With the development of the large language models (LLM), Question Answering (QA) has become one of the main forms of interaction between humans and models, making the study of QA data increasingly highlight. We focus on QA data of Machine Reading Comprehension (MRC), which plays a crucial role in developing QA systems and assessing models' multidimensional understanding abilities.

Various approaches have been proposed to address the challenge of data scarcity in MRC through the synthesis of domain-specific QA pairs using neural question generation (QG) models (Golub et al., 2017; Wang et al., 2019; Shakeri et al., 2020; Puri et al., 2020; Lee et al., 2020; Yue et al., 2021; Yao et al., 2022). However, generated QA pairs are often of low quality and cause harm rather than improvement when all added into training data. This brings forward a crucial selection problem: given a set of synthetic QA pairs, how to select high-quality ones from all candidate data that are beneficial for future applications.

Previous selection methods can be classified into internal and external approaches. Internal methods utilize the likelihood estimation, known as LM Score, obtained from the QG model for selection (Shakeri et al., 2020). External methods, on the other hand, employ an external model such as a pre-trained QA model (Alberti et al., 2019), a rank classifier (Yao et al., 2022) or a question value estimator (Yue et al., 2022) for selection. While these methods have shown some improvement in downstream QA

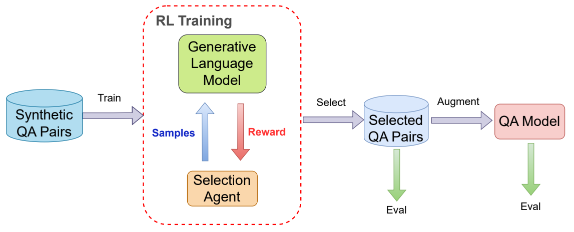

Figure 1: Illustration of our method. A selection agent is trained under reward guidance from LLM based on synthetic QA pairs, and select high-quality QA pairs from the same synthetic QA pairs. The selected QA pairs are augmented into the training of QA model.

performance (Rennie et al., 2020; Yue et al., 2022), there is still potential for enhancing the selection outcomes. And these methods predominantly focus on specific metrics of the downstream QA task and lack generalization across different metrics. Consequently, QA pairs selected through these methods may be beneficial for specific metrics exclusively. The selection criteria are inflexible and challenging to tailor to varying requirements of downstream tasks. Moreover, the selection process lacks intuitive explanations, failing to offer insights into the distinguishing attributes of the selected QA pairs.

In this paper, we propose to apply generative large pre-trained language model (LLM) for flexible and interpretable selection. We train a selection agent which takes in synthetic QA pairs and outputs real-valued scores as an inner evaluation metric for their qualities. The inherent interpretation of scores can be flexible and adapts to diverse downstream tasks and metrics. Due to the lack of supervision and inspired by Yue et al. (2022), we use Reinforcement Learning (RL) algorithm to train the selection agent. LLM have demonstrated promising capabilities in quality assessment and reward design (Fu et al., 2023; Kwon et al., 2023). Therefore, we apply LLM to provide reward and guide the training process. Figure 1 illustrates the overall framework.

To comprehensively evaluate our method, we consider both generative and extractive datasets, following the experimental setup of Yao et al. (2022) and Yue et al. (2022) respectively, to ensure a fair comparison with previous works. Experimental results show that our method outperforms existing selection methods in downstream performance. We also experiment with LLM as a selector and QA model, finding it performs best as a reward model. And we use the competency assessment framework proposed by Wang et al. (2022) to show the multi-dimensional high quality of our selected data compared to other methods. Our method has flexibility across metrics and provides interpretability for the selected QA pairs. Our contributions are as follows:

- We propose to apply generative large pre-trained language model (LLM) as reward model to train a selection agent with RL, which can effectively select high-quality examples to augment the training data in MRC.
- We conduct experiments on both generative and extractive datasets, and the results show that our selection agent leads to superior downstream performance compared to other selection methods independent of dataset type.
- We assess quality of selected data from multiple dimensions, and our method can select QA pairs with higher competency values in most dimensions. Additionally, our method extends well to example selection for constructing demonstrations on few-shot setting, yielding notable results with limited examples.

2. Related Work

Synthetic QA pairs The synthesis of QA pairs from unlabeled contexts has emerged as a significant research area, involving the generation of questions and corresponding answers. The generation of questions and answers can be related (Shakeri et al., 2020; Lee et al., 2020; Yue et al., 2021; Yao et al., 2022; Yue et al., 2022) or independent of each other (Alberti et al., 2019; Puri et al., 2020). With the development of Question Generation (QG) techniques (Du et al., 2017; Sun et al., 2018; Nema et al., 2019), a commonly adopted approach is to leverage a QG model to generate questions based on pre-processed answers. The answers can either be identical to those in original datasets (Yue et al., 2022), or extracted from unlabeled text with special methods such as heuristics-based methods (Yao et al., 2022), an classifier (Puri et al., 2020) or a seq2seq model (Shakeri et al., 2020). We focus on the selection of synthetic QA pairs rather than their generation. Therefore, we followed Yue et al. (2022) to synthesize QA pairs in extractive datasets and followed Yao et al. (2022) to synthesize QA pairs in generative datasets. This allows us to encompass various types of data and establish a solid basis for fair analysis.

Data Selection How to select valuable samples from all given data has been widely studied in machine learning community as active learning (Settles, 2009; Ebrahimi et al., 2020). As for QA pairs, data selection is an internal part of the QA pair generation pipeline. Previous selection methods can be divided into two categories based on whether external models are used in the process. The internal methods use generation log likelihood generated by the QG model, known as LM score(Shakeri et al., 2020), as a metric for selection, which is intuitive but not performant enough. The external methods use an external model for ranking, such as a pre-trained QA model (Alberti et al., 2019), a pre-trained ranking classifier (Yao et al., 2022), a question value estimator (Yue et al., 2022). Among these, using question value estimator trained by RL has made impressive performance in semi-supervised domain adaption of QA (Yue et al., 2022). But they primarily focus on downstream metrics, lack generalization across different metrics, and fail to reflect the characteristic of selected QA pairs. Furthermore, they incur high computational costs during RL training. We propose to apply generative large pre-trained language models as reward model to train a selection agent with RL, which offers the advantages of flexible adaptation to various metrics and datasets, improves interpretability, and reduces training expenses.

Apply LLM in Reinforcement Learning The employment of RL in NLP research is an increasingly researched issue, especially with the development of large pretrained language models(LLM). RL can be used to help the training and alignment of LLM (Ouyang et al., 2022; Pang et al., 2023), while LLM can be used as reward model to guide RL training due to their increasing abilities (Kwon et al., 2023; Rafailov et al., 2023). Fu et al. (2023) propose to utilize the emergent abilities (e.g., zero-shot instruction) of generative pre-trained models to score generated texts. We apply generative language model to assess synthetic QA pairs in a similar way and combine it with RL algorithm.

3. Method

3.1. Background and Notations

First, we describe the task with notations formally. We denote the original QA dataset as $\mathcal{D}^{\text{ori}} = \{c_i^{\text{ori}}, q_i^{\text{ori}}, a_i^{\text{ori}}\}_{i=1}^{N}$, where c, q, a means context, question and answer correspondingly, N is the size of original QA dataset. Considering that original QA dataset is not sufficient, synthetic QA pairs are required. Synthetic QA pairs are generated mainly through a QG model, which is trained and used differently in generative and extractive datasets (see details in section 4). We denote the synthetic QA pairs as $\mathcal{D}^{\text{syn}} = \{c_i^{\text{syn}}, q_i^{\text{syn}}, a_i^{\text{syn}}\}$ regardless of its type and generation method.

Our task is to select high-quality ones from $\mathcal{D}^{\text{syn}}$ that are not only helpful for enhancing downstream metrics, but also show great data quality from multiple angles.

3.2. Selection Agent

A selection decision is often made based on some scores that can indicate the quality of QA pairs (Yue et al., 2022). The score function g take in a synthetic QA sample and output its corresponding score v, i.e.,$v_i = g(c_i^{syn}, q_i^{syn}, a_i^{syn})$. The implementation of g varies as we described in section 2.

We adopt BERT model as selection agent's backbone as in Yue et al. (2022) for a fair comparison of different training methods. However, they incorporated dataset-specific information as additional features in the hidden representation, which is applicable only to extractive datasets and not to generative datasets. To build a general dataset-independent selection agent, we simplify the structure to BERT model and linear classifier layers, described as follows:

$$\mathbf{h} = \text{BERT}\ [\texttt{<CLS>}\ q_i^{syn}\ \texttt{<ANS>}\ a_i^{syn}\ \texttt{<SEP>}\ c_i^{syn}]$$
$$\mathbf{h}' = \sigma(W_2\sigma(W_1\mathbf{h} + b_1) + b_2)$$
$$v_i = W_3\mathbf{h}' + b_3$$

where `<CLS>`, `<ANS>`, `<SEP>` are special delimiter tokens, $\mathbf{h} \in \mathbb{R}^H$ is representation derived from `<CLS>`, $W_1 \in \mathbb{R}^{H_1 \times H}, b_1 \in \mathbb{R}^{H_1}, W_2 \in \mathbb{R}^{H_2 \times H_1}, b_2 \in \mathbb{R}^{H_2}, W_3 \in \mathbb{R}^{H_2}, b_3 \in \mathbb{R}$ are trainable parameters from linear layers[1], σ is the activation function `tanh`. The whole process can presented as $v_i = e_\gamma(q_i^{syn}, a_i^{syn}, c_i^{syn})$, where e_γ represents the selection agent.

[1]See appendix B for more details.

3.3. Agent Training: Reward from LLM

We can't train the selection agent with supervised methods due to the lack of labeled data on the true values of synthetic QA pairs. Inspired by Yue et al. (2022), we use RL to train the selection agent.

From the perspective of RL, actions refer to whether to select a certain QA pair, states refer to the selected data, state space increases exponentially with the number of datas, and the rewards are used to estimate the states (i.e., currently selected examples). The actions are related to values given by selection agent. We adopt the Bernoulli sampling based on the values as policy, which encourages the policy exploration compared to strict threshold. We followed Yue et al. (2022) to formulate the decision making policy π_γ in batch-wise fashion:

$$v_i = e_\gamma(q_i^{syn}, a_i^{syn}, c_i^{syn})$$
$$s_i \sim \texttt{Bernoulli}(v_i)$$
$$\pi_\gamma(\mathcal{S}|\mathcal{D}_B^{syn}) = \prod_{i=1}^{B}[v_i^{s_i} \cdot (1 - v_i)^{1-s_i}]$$

where e_γ is the selection agent, B is the batch size, $\mathcal{D}_B^{syn}$ is the batch data and $\mathcal{S} = \{s_0, s_1, ..., s_B\}$ is the selection vector, $s_i \in \{0, 1\}$ for $i \in [1, B]$.

The selection agent is trained to maximize the rewards and is updated through REINFORCE algorithm (Williams, 1992).

$$r = \texttt{reward\_fn}(\mathcal{S}, \mathcal{D}_B^{syn})$$
$$\mathcal{L}_\gamma = -\mathbb{E}_{\mathcal{S}\sim\pi_\gamma(\cdot|\mathcal{D}_B^{syn})}[r]$$
$$\nabla_\gamma\mathcal{L}_\gamma = -\mathbb{E}_{\mathcal{S}\sim\pi_\gamma}[r\nabla_\gamma \log \pi_\gamma(\mathcal{S}|\mathcal{D}_B^{syn})]$$
$$= -\mathbb{E}_{\mathcal{S}\sim\pi_\gamma}[r\nabla_\gamma \sum_{i=1}^{B} \log[v_i^{s_i}(1 - v_i)^{1-s_i}]]$$

where r is the reward of current selection, $\mathcal{L}_\gamma$ is the loss function that needs to be minimized and $\nabla_\gamma\mathcal{L}_\gamma$ is the gradient of selection agent with RL training.

One of the significant distinctions we introduce in our method is the implementation of reward function. Instead of performance-driven reward which is rigid and costly, we use generative large language models (LLM) as reward model to generate reward. The motivation is that the high-quality data tend to receive higher probability based on given instructions and contexts within generative language models, which has been shown in many text generation tasks, including dialogue response, text summarization, data-to-text and machine translation (Fu et al., 2023). We explore this characteristic in synthetic QA pairs.

The reward is calculated as follows. The input of reward function is the selection vector $\mathcal{S}$ and the batch data $\mathcal{D}_B^{syn}$ generated by the selection agent. We first extract current selected data $\mathcal{D}_S^{syn}$

PromptTemplate		
Type	**Instruction based T**	**Answer A**
SEM_1	Answer the question based on the conversation between a human and AI.\nQuestion: Are the responses of AI semantically appropriate? (a) Yes. (b) No.\nConversation: Human asked, {context}. {question} AI answered {answer}	Yes.
SEM_2	Answer the question based on the conversation between a human and AI.\nQuestion: Are the responses of AI semantically appropriate? (a) Yes. (b) No.\nConversation: Human asked, {context}. Based on the context, {question} AI answered {answer}	Yes.
SEM_3	Answer the question based on the conversation between a human and AI.\nQuestion: Are the responses of AI semantically appropriate? (a) Yes. (b) No.\nConversation: Human said, answer the question based on the given text. The question is {question} The text is {context}. AI answered {answer}	Yes.
QUE	Answer the question based on the conversation between a human and AI.\nQuestion: Is the question generated by AI related to the conversation? (a) Yes. (b) No.\nConversation:Human said, generate a question based on the following text. {context}. AI said {question}	Yes.
MRC	Answer the question based on the given context.\nQuestion: {question}\nContext: {context}	{answer}.

Table 1: The design of PromptTemplate is based on instructions from Fu et al. (2023), and the output is T and A correspondingly. SEM_1,SEM_2,SEM_3, QUE construct the prompting template from dialogue-level, while MRC is from pure MRC-level. SEM_1 is the default.

from $\mathcal{D}_B^{syn}$ based on $\mathcal{S}$: $\mathcal{D}_S^{syn} = \{(q_i^{syn}, a_i^{syn}, c_i^{syn}) \mid (q_i^{syn}, a_i^{syn}, c_i^{syn}) \in \mathcal{D}_B^{syn} \wedge s_i = 1\}$. For each sample in $\mathcal{D}_S^{syn}$, we reconstruct it using a special prompt template and compute the overall conditional probability of the generative language model. And we use the average of all rewards of $\mathcal{D}_S^{syn}$ as the final reward. The whole process is formally described as follows:

$$A^i, T^i = \texttt{PromptTemplate}(q_i^{syn}, a_i^{syn}, c_i^{syn})$$

$$r_i = \sum_{k=1}^{m} \log p_\theta(A_k^i \mid A_{<k}^i, T^i, \theta)$$

$$r_{\mathcal{S}} = \frac{\sum_i r_i}{\mid \mathcal{D}_S^{syn} \mid}$$

where `PromptTemplate` is the instruction-based template used to construct prompt text, A^i and T^i are the target text and prompt text constructed from template, A_k^i is the k-th token of A^i, m is the number of tokens of A^i, and θ is the parameters of the generative language model. The details and candidates of `PromptTemplate` is represented in table 1. We use $r_{\mathcal{S}}$ to train and update the selection agent. The complete training process is shown in appendix A in the form of pseudocode.

Due to the diversity of instructions and the interpretability of probabilities, we believe that this training method is more flexible and interpratable than performance-driven reward method.

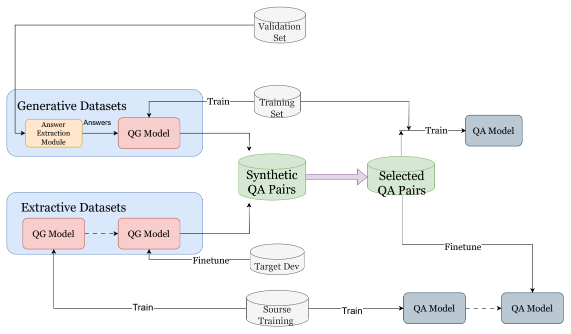

Figure 2: Illustration of our experiments. We consider both generative and extractive datasets, using different processes to generate synthetic QA pairs. See section 4.2 and 4.3 for more details.

4. Experiments

4.1. Experimental Setup

The details about datasets are different in different types of datasets, which are described in the following sections. But the whole process remains consistent. We first employ the provided data to train a QG model, then use the pre-trained QG model to generate questions and construct QA pairs. These synthetic QA pairs serve as the basis for applying various selection methods to select high-quality samples. The selected data is incorporated into the training of a QA model. The performance of the QA model on test sets provides insights into the quality of the selected data. The entire process

	Different Selection Methods		
Dataset	ALL	RTC	Other
FairytaleQA	9726	-	5835
NQ	104,071	53377	62,442
TriviaQA	61,688	25389	37,012

Table 2: Number of synthetic datas selected by different methods. Other is our method and any selection method in section 4.1 except RTC. RTC is not suitable for generative dataset.

is shown in the figure 2 for better illustration.

The models are all implemented using Hugging Face transformers library (Wolf et al., 2019). In order to conduct a fair and comprehensive comparison, we implement the following baselines: **(1) No Augmentation**: we train the QA model with original train dataset without any data augmentation. **(2) Dev as Augmentation**: We use data of the validation dataset as data augmentation to train the QA model. Data from the validation dataset is representative of high-quality data, but is scarce. **(3) All Augmentation**: we train the QA model with original given train dataset and all synthetic QA pairs. **(4) Random Selection**: we random select synthetic QA pairs from all generated data and use them in the training. **(5) LM Filtering (Shakeri et al., 2020)**: we use the generation log likelihood (LM Score) generated by the QG model as metrics to rank the synthetic QA pairs and keep the top K% as data augmentation. **(6) RTC Selection (Alberti et al., 2019)**: we use the pretrained QA model in (2) to select QA pairs that are solvable from all synthetic data. This baseline is only used in extractive dataset because the threshold of solvable samples is vague for generative datasets. **(7) QVE(Question Value Estimator, Yue et al. (2022))**: This method applies direct QA performance as reward to train a estimator to select QA pairs. We reproduce their work. Note that we apply marginal information in the extractive dataset but not in the generative datasets (See section 3.2 for more details). **(8) Rank Model**: This method is proposed by Yao et al. (2022), which trains a external classifier to select QA pairs. Since the technical details of training is not public, we just use their released model to select data. This baseline is only used in the generation dataset. **(9) LLM as Selectors**: We use the LLM directly for selection based on the reward they generated. This baseline can reflect the effect of RL framework compared to our method.

In our approach, we mainly consider Llama2-7b and Llama2-13b (Touvron et al., 2023) as our reward model due to cost consideration and their impressive performances, but the reward model can be extended to other generative language models as well. For the same data type, candidate synthetic QA pairs are identical for all selection methods for fair comparison. However, the data generation process, the training method of QA model and evaluation metrics are different for different data types, which are described in the following sections. To mitigate the impact of the amount of training data on downstream performance, we make the number of output selected QA pairs same for all selection methods except RTC, as shown in table 2. The numbers of selected QA pairs of our method are the same as "Other" in the table 2.

No.	Methods	Rouge-L
(1)	No Augmentation	52.98
(2)	Dev as Augmentation	53.04
(3)	All Augmentation	51.39
(4)	Random Selection	52.62
(5)	LM Filtering	50.24
(7)	QVE	51.6
(8)	Rand Model	49.9
(9)	As selector	
	llama2-7b	51.64
	llama2-13b	52.57
Our	As reward model	
	llama2-7b	53.32
	llama2-13b	**53.44**

Table 3: Generative performance of FairytaleQA test set on data selected by different methods.

To eliminate the impact of randomization variance, all experimental results in this paper are obtained based on the average of three different seeds.

4.2. Generative MRC Dataset

4.2.1. Dataset

Following Yao et al. (2022), we use FairytaleQA (Xu et al., 2022) dataset for experiments. FairytaleQA dataset is a generative MRC datatset released in recent years, which focuses on educational narrative comprehension and covers seven types of narrative elements or relations.

4.2.2. Training Details and Metrics

Yao et al. (2022) proposed a pipeline for question-answer generation (QAG) based on FairytaleQA dataset. The pipeline consists of three steps: answer extraction, question generation and neural network based ranking. We follow the first two steps to generate synthetic QA pairs.

The backbone of both QG model and QA model is `BART-Large` for fair comparison. We use all labeled data of training dataset to train the QG model. The answers are extracted based on heuristics-based AG Module proposed by Yao et al. (2022). And we use pretrained QG model and extracted

No.	Methods	NQ		TriviaQA	
		EM	F1	EM	F1
(1)	No Augmentation, Source only	43.31	57.92	48.89	58.29
(2)	Dev as Augmentation	52.94	67.01	58.02	63.89
(3)	All Augmentation	59.75	71.99	61.89	66.89
(4)	Random Selection	58.57	70.99	59.97	65.57
(5)	LM Filtering(Shakeri et al., 2020)	57.84	70.57	60.69	66.09
(6)	RTC Selection(Alberti et al., 2019)	58.43	70.84	60.76	66.05
(7)	QVE with marginal info (Yue et al., 2022)	58.82	71.15	60.73	66.12
(9)	As selector				
	llama2-7b	58.6	71.17	60.75	66.20
	llama2-13b	58.84	71.32	61.07	66.48
Our	As reward model				
	llama2-7b	59.05	71.61	61.13	66.39
	llama2-13b	**59.24**	**71.67**	**61.82**	**67.17**

Table 4: Semi-supervised domain adaptation performance on data selected by different selection methods on extractive datasets.

answers to generate questions in the context of original validation dataset and construct synthetic QA pairs. The selected QA pairs will be mixed into the training data to train the QA model. See appendix B for more details about hyperparameters.

The performance of the QA model is evaluated through Rouge-L value on test dataset like Yao et al. (2022), which reflects the downstream performance of different selection methods.

4.2.3. Results

Table 3 shows the overall results of all baselines and our method in generative MRC datasets.

The results show that: (1) Including all synthetic QA pairs in the training data has a detrimental effect on the original performance, indicating the low overall quality of the synthetic data. (2) Random selection achieves better performance than most other selection methods, indicating that most other methods struggle if the overall quality of candidates is low. (3) Among all selection methods, our method achieves the best performance. The subpar performance of the rank model (No.8) proposed by Yao et al. (2022) may be due to the fact that the published model is not suitable for our setting and environment. (4) Applying LLM as a reward model rather than a selector can introduce more improvements, which means using RL to train selection agents is better than reward-based direct ranking, and even outperforms ranking model with larger scale (llama-7b as reward model outperforms llama2-13b as selector). The reason might be that the rewards generated by LLM do not directly align with the evaluation metrics, or rewards are not fully reliable due to limited model size. Hence, better performance can be obtained where policy exploration is encouraged in RL.

4.3. Extractive MRC Dataset

4.3.1. Dataset

To make a fair comparison with Yue et al. (2022), we set our experiments as semi-supervised domain adaption of QA and assume only a small number of target annotations are available. We use SQuAD 1.1 (Rajpurkar et al., 2016) as source datasets, NaturalQuestions (NQ) (Kwiatkowski et al., 2019) and TriviaQA (Joshi et al., 2017) as target datasets.

4.3.2. Training Details and Metrics

For each target dataset, the original validation dataset is used as the test set, all context and only a small number of labeled QA pairs of training dataset are available for training (1000 samples, about 1%-1.5%). And during training, another small amount of labeled data is extracted from the training data set as a verification data set, which we define as a sample development set for illustration.

The backbones of QG model and QA model are `BART-base` (Lewis et al., 2020) and `BERT-base-uncased` (Devlin et al., 2019) respectively. The QG model is trained with source training datasets and then finetuned on the sample dev set. Then we use the pre-trained QG model to generate questions on all the context of training datasets based on original answers. The generated questions are combined with original answers to form QA pairs.

We apply LLM for reward generation, as described in section 3. But the context of MRC data is often very long and may exceed the maximum context length of language models. To address this issue, we split the contexts into chunks and calculate the reward separately, taking the maximum value of the rewards in different chunks as the reward for this sample. See appendix B for more details about hyperparameters.

QA Model	Setting	FairytaleQA	NQ		TriviaQA	
		Rouge-L	EM	F1	EM	F1
llama2-7b	0-shot	15.78	29.00	42.40	29.33	33.98
	Max-shot	30.44	45.83	59.37	53.23	61.67
llama2-13b	0-shot	23.19	32.89	45.52	30.93	44.21
	Max-shot	42.93	53.61	66.90	57.50	65.79

Table 5: Performance on the same test set as previous with LLM as QA model under zero-shot and few-shot setting. Max-shot refers to using as many examples as possible within context limitation to construct the demonstration.

The evaluation metrics is QA performance (EM and F1) on test set, where QA model is trained with source dataset and then finetuned on target data.

4.3.3. Results

Table 4 shows the overall results of all baselines and our method in extractive MRC dataset.

The results show that: (1) Using all of synthetic QA pairs in the training yields the best performance due to large data volume. This also indicates that the overall quality of synthetic data is high. (2) Among all selection methods, QVE and using LLM as selector are more effective compared to random selection, LM and RTC methods. Note that the different results compared to Yue et al. (2022) may be attributed to different devices, transformers version or hyperparameters like batch size[2]. Due to the emergent ability of LLM and its implicit familiarity with Wikipedia texts, using LLM as selector directly can induce comparable and even better performance than previous selection methods. This phenomenon was not found on generative dataset, and we assume it is attributed to the relatively high overall quality of synthesis QA pairs, as said in section 4.2. (3) Our method outperforms all other selection methods, achieving performance comparable to using all data with only 60% of the data. We find that applying large language model as reward outperforms using it as selector for the same type of models, as said in generative dataset.

5. Analysis

5.1. LLM as QA model

We propose to use LLM as reward model to select high-quality QA pairs to enhance the downstream QA performance, and we have made experiments to demonstrate that our method is better than using LLM directly as selector. But from the perspective of QA system, LLM has the potential to be used as QA model to enhance the downstream performance. Therefore, we make experiments on the same datasets as in section 4.2 and 4.3 to show the performance of LLM as QA model rather than selecting examples.

[2]These objective hardware factors have an impact on both the selection process and the generation process of candidate QA pairs, which lead to performance gap with original paper (Yue et al., 2022)

QA Model	Method	FairytaleQA
llama2-7b	random	26.26
	llama2-7b (selector)	28.5
	llama2-7b (reward)	**30.48**

Table 6: 5-shot Performance (Rouge-L) on the Fairytale test set. Method refers to the method for selecting examples from validation set to construct demonstrations.

Since the fine-tuning process of llama2-7b and llama2-13b is very hardware-consuming and our experimental scenarios are based on limited supervised data, we conduct the experiments under zero-shot and few-shot settings. The test datasets are the same as in section 4.2 and 4.3. For few-shot setting, we random select examples from corresponding validation datasets to construct demonstrations. To mitigate the influence of prompts, we directly use prompts from widely adopted PromptSource (Bach et al., 2022).

The results are presented in table 5. "Max-shot" indicates that using as many examples as possible within limited context window length of each model. The number of examples used in the demonstrations of each test sample is dynamically determined by the length of test sample, randomly chosen examples and context window.

The results show that more examples in the demonstration can bring more help to LLM as QA model. And the larger the model, the better the performance, which is consistent with the community consensus. With the scale of 7b and 13b, the performance of LLM as QA model still lags behind previous results in table 3 and table 4. This indicates that LLM can serve as a reward model rather than a selector or QA model to bring more assistance to QA systems. However, the results of llama2-13b are close to the previous results, and using a larger model as QA model may exceed the previous results. Therefore, we make further experiments to

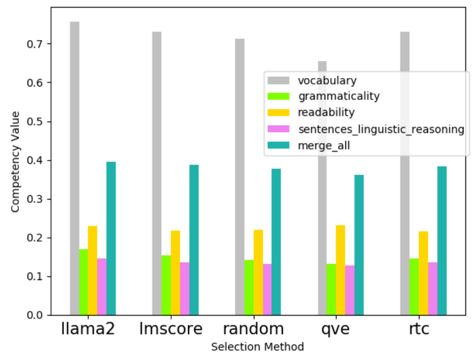

Figure 3: Values of selected NQ data in competency assessment framework. The higher the value, the better the quality of corresponding dimension. Our method with llama2-7b is shown as 'llama2'.

show that our selecting method can also be helpful for few-shot performance by selecting high-quality examples to construct demonstrations.

We choose FairytaleQA dataset as representative due to its relative difficulty. We test llama2-7b model under few-shot setting with 5 examples selected from validation dataset to construct demonstrations. The selection methods we considered here are random, LLM as selector, LLM as reward model since other methods are not suitable. The results are presented in the table 6. The results show that our method is not limited to selecting synthesis data, but is applicable to selecting demonstration examples. Using llama-7b as QA model with 5 examples selected by our method can outperform the "Max-shot" setting. Based on the random results from table 6, we can infer that the number of examples for "Max-shot" exceeds 5. Therefore, even if using a larger LLM as QA model can achieve better results, our method can still be further improved on this basis.

5.2. Analysis on Selected Examples

Experimental results in section 4.2 and 4.3 show that our selection agent helps QA model to reach better performance in comparison with existing selection methods. To further evaluate quality of selected QA pairs of different methods, we take selected NQ data as examples and use competency assessment framework proposed by Wang et al. (2022) to assesses data property from multiple dimensions, including vocabulary, grammaticality, readability, sentences linguistics reasoning, and merge all. See original paper for more details. Due to the incompleteness of their public code, some dimensions are not assessed in our work. We use llama2-7b model as reward model to represent our method.

The results is presented in figure 3. The higher the column value, the better the quality for each dimension. According to Wang et al. (2022), the samples with high values of certain aspect can be used in helping model improve corresponding capability boundary. The results demonstrate that our method selects high-quality data with high competency values in most dimensions compared to other selection methods, which is helpful for extending model's capability boundary. The results also show that qve method with performance-driven reward lacks generalization across different metrics.

Prompt Type	NQ	
	EM	F1
SEM_1(default)	**59.05**	**71.61**
SEM_2	58.91	70.96
SEM_3	58.75	71.28
QUE + SEM_1	58.97	71.40
MRC	57.83	70.92

Table 7: Impact of different prompting templates on NQ performance with llama2-7b as reward model. The rest settings are the same as table 4.

5.3. Flexibility and Interpretability

In section 3, we have provided a theoretical explanation of the flexibility and interpretability features inherent in our method. In this section, we illustrate through experiments and demonstrations.

As for flexibility, prompt templates have multiple formats as shown in table 1. Due to the inherent flexibility of prompting, our method can be easily applied in different metrics and tasks. For example, if the evaluation metric for selected data is about grammar, we can easily change the prompt template to *Are the responses of AI grammatically correct?* Then the selection agent that trained with this template tends to select data with strict grammar. Therefore, our method is flexible and easy to apply to different metrics.

In addition, we explore the effect of different prompt templates for similar tasks listed in the table 1, which can inform the choice of prompt template.

We choose llama2-7b model as reward model and test in NQ as before, and list the results in table 7. The results show that different templates result in different performance, but most are still superior or comparable to other selection methods in table 4. This also indicates that the format of dialogue is more helpful than MRC for our task. The more concise the template is, the better the performance.

As for interpretability, we show it through the actual meaning of scores given by selection agent. Most selection methods filter based on their scores, but they can't explain why this sample with high scores and others not. However, our score has actual meaning. For examples, if a sample is scored

RL step	NQ	
	EM	F1
step = 0 (direct)	58.14	70.44
step = 1000	58.59	70.83
step = 2000	58.79	71.31
step = 3000	**59.05**	**71.61**
step = 4000	58.84	71.16
step = 5000	58.37	70.7

Table 8: Impact of different RL training steps on NQ performance with llama2-7b as reward model.

0.7 by selection agent trained with default template, it indicates that the estimated probability of this sample's answer semantically matching its question and context is about 0.7. Based on prompting template, it becomes evident that the differences in sample values offer insights into the distinctive features encapsulated within the selected QA pairs.

5.4. The Influence of RL

In addition to regular factors as seed, batch size and learning rate, we conduct additional experiments on RL training steps to show its influence.

We set the step size to 0 to directly use the untrained agent model (which is BERT in our experiments) to rank and select QA pairs. The rest indicate that we use the selection agent trained with RL for the corresponding steps to make the selections. We show the results in table 8.

The results show that training steps significantly impact the performance, which brings forward the issue about the convergence of RL training. The performance of the selection agent trained with few steps tends to be stochastic, while training with a large number of steps may lead to overfitting or forgetting. The overall performance presents a form of increasing first and then decreasing, and the optimal training steps is the peak point. We found the same issue when we reproduced Yue et al. (2022)'s work. Finding the optimal training steps is crucial for great performance. All the previously reported results in this paper are obtained using the optimal number of training steps.[3]

6. Conclusion and Future Work

We propose to train the selection agent with generative language model as reward in RL framework, which can select high-quality synthetic QA pairs independent of downstream metrics. Our method outperforms existing selection methods in downstream performance on both generative and extractive datasets, while offering flexibility across metrics and interpretability for the selected QA pairs. The experimental results also show that using LLM as a reward model is better than using it as a direct selector or QA model in our task.

In this paper, our focus has been on generative language models with 7b and 13b due to cost considerations. However, large language models exhibit impressive emergent abilities that may aid our task and are good reward models, which is still need to be explored. And our work focus on the selection process only, assuming the availability of a pool of synthetic QA pairs. We will attempt to combine our core idea into the generation process of synthetic QA pairs and generate high-quality data.

In conclusion, our method demonstrates an instance to use large language model in MRC and explore its combination with RL. Due to the diversity of prompt template, our work can be easily extended to other tasks beyond QA. In the future, we will further explore more language models and their combination with RL in more tasks and applications.

[3]We set the training steps to a large number and saved checkpoints with different training steps. The checkpoint with the highest F1 score is identified as the optimal training step, ensuring the selection of the most effective model. To ensure a fair comparison, we applied a similar approach to confirm the optimal steps for other baselines.

7. Acknowledgements

This work was supported by National Science and Technology Major Project (2022ZD0116308) and National Natural Science Foundation of China (62036001). We also appreciate the anonymous reviewers for their valuable suggestions. The corresponding author is Houfeng Wang.

8. Bibliographical References

Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. 2019. Synthetic qa corpora generation with roundtrip consistency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6168–6173.

Stephen H. Bach, Victor Sanh, Zheng-Xin Yong, Albert Webson, Colin Raffel, Nihal V. Nayak, Abheesht Sharma, Taewoon Kim, M Saiful Bari, Thibault Fevry, Zaid Alyafeai, Manan Dey, Andrea Santilli, Zhiqing Sun, Srulik Ben-David, Canwen Xu, Gunjan Chhablani, Han Wang, Jason Alan Fries, Maged S. Al-shaibani, Shanya Sharma, Urmish Thakker, Khalid Almubarak, Xiangru Tang, Xiangru Tang, Mike Tian-Jian Jiang, and Alexander M. Rush. 2022. Promptsource: An integrated development environment and repository for natural language prompts.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1342–1352.

Sayna Ebrahimi, William Gan, Dian Chen, Giscard Biamby, Kamyar Salahi, Michael Laielli, Shizhan Zhu, and Trevor Darrell. 2020. Minimax active learning. *arXiv preprint arXiv:2012.10467*.

Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. 2023. Gptscore: Evaluate as you desire. *arXiv preprint arXiv:2302.04166*.

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

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

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur 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. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. 2023. Reward design with language models. *arXiv preprint arXiv:2303.00001*.

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 *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 208–224.

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 *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

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 *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3314–3323.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Jing-Cheng Pang, Pengyuan Wang, Kaiyuan Li, Xiong-Hui Chen, Jiacheng Xu, Zongzhang Zhang, and Yang Yu. 2023. Language model self-improvement by reinforcement learning contemplation. *arXiv preprint arXiv:2305.14483*.

Raul Puri, Ryan Spring, Mohammad Shoeybi, Mostofa Patwary, and Bryan Catanzaro. 2020. Training question answering models from synthetic data. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5811–5826.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*.

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

Steven Rennie, Etienne Marcheret, Neil Mallinar, David Nahamoo, and Vaibhava Goel. 2020. Unsupervised adaptation of question answering systems via generative self-training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1148–1157.

Burr Settles. 2009. Active learning literature survey.

Siamak Shakeri, Cicero 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 *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5445–5460.

Xingwu Sun, Jing Liu, Yajuan Lyu, Wei He, Yanjun Ma, and Shi Wang. 2018. Answer-focused and position-aware neural question generation. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 3930–3939.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Huazheng Wang, Zhe Gan, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, and Hongning Wang. 2019. Adversarial domain adaptation for machine reading comprehension. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2510–2520.

Xiaoqiang Wang, Bang Liu, Fangli Xu, Bo Long, Siliang Tang, and Lingfei Wu. 2022. Feeding what you need by understanding what you learned. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5858–5874.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Reinforcement learning*, pages 5–32.

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. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Ying Xu, Dakuo Wang, Mo Yu, Daniel Ritchie, Bingsheng Yao, Tongshuang Wu, Zheng Zhang, Toby Li, Nora Bradford, Branda Sun, Tran Hoang, Yisi Sang, Yufang Hou, Xiaojuan Ma, Diyi Yang, Nanyun Peng, Zhou Yu, and Mark Warschauer. 2022. Fantastic questions and where to find them: FairytaleQA – an authentic dataset for narrative comprehension. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 447–460, Dublin, Ireland. Association for Computational Linguistics.

Bingsheng Yao, Dakuo Wang, Tongshuang Wu, Zheng Zhang, Toby Li, Mo Yu, and Ying Xu. 2022. It is ai's turn to ask humans a question: Question-answer pair generation for children's story books. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 731–744.

Xiang Yue, Ziyu Yao, and Huan Sun. 2022. Synthetic question value estimation for domain adaptation of question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*.

Zhenrui Yue, Bernhard Kratzwald, and Stefan Feuerriegel. 2021. Contrastive domain adaptation for question answering using limited text corpora. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9575–9593.

A. Pseudocode of Training Algorithm

We propose to apply generative language model as reward to train the selection agent with RL, which has described in section 3.3. The whole process can be described in the format of pseudocode in Algorithm 1, and the reward function is described in Algorithm 2.

B. Hyperparameters

In this section, we list the hyperparameters used in the experiments in section 4.

According to section 3 and appendix A, the required hyperparameters during the RL training of selection agent are as follows: batch size B, training step T,learning rate of the selection agent α, chunk length C, hidden layer dimension H_1,H_2,H_3. In addition to the training process, training the QG model to generate questions and training the QA model to evaluate using metrics from downstream tasks also requires some hyperparameters such as learning rate, epoch, max_seq_length and so on. For a fair comparison, we keep these hyperparameters as Yue et al. (2022) and Yao et al. (2022).

Algorithm 1 The Selection Agent Training Method

```
Input: generative language model $M$; synthetic QA pairs $D^{syn}$;
Hyperparameters: batch size $B$, training step $T$, learning rate of selection agent $\alpha$.
Output: trained selection agent $e_\gamma$.
1: Randomly initialize $e_\gamma$
2: for step $= 1$ to $T$ do
3:     ▷ ① Sample a batch of synthetic QA pairs:
4:     Sample $\mathcal{D}_B^{syn} = \{(c_i^{syn}, q_i^{syn}, a_i^{syn})\}_{i=1}^B$ from $D^{syn}$
5:     ▷ ② Select with selection agent:
6:     $\mathcal{V} = e_\gamma(\mathcal{D}_B^{syn})$
7:     ▷ ③ Sample selection vector:
8:     $\mathcal{S} \sim \mathtt{Bernoulli}(\mathcal{V})$
9:     ▷ ④ Get reward from $M$:
10:    $r = \mathtt{reward\_fn}_M(\mathcal{S}, \mathcal{D}_B^{syn})$
11:    ▷ ⑤ Update the selection agent:
12:    $\mathcal{L}_\gamma = -\mathbb{E}_{\mathcal{S}\sim\pi_\gamma(\cdot|\mathcal{D}_B^{syn})}[r]$
13:    $\nabla_\gamma \mathcal{L}_\gamma = -\mathbb{E}_{\mathcal{S}\sim\pi_\gamma}[r\nabla_\gamma \log \pi_\gamma(\mathcal{S}|\mathcal{D}_B^{syn})]$
14:    $\gamma \leftarrow \gamma - \alpha \cdot \nabla_\gamma \mathcal{L}_\gamma$
15: end for
16: return $e_\gamma$
```

Algorithm 2 Reward function based on generative language model

```
Input: selection vector $\mathcal{S}$, candidate synthetic QA pairs $\mathcal{D}_B^{syn}$;
Hyperparameters: generative language model $M$ and its parameters $\theta_M$.
Output: reward.
1: for i $= 1$ to $B$ do
2:     ▷ ① Construct the prompt:
3:     $A^l, T^l = \mathtt{PromptTemp}(q_i^{syn}, a_i^{syn}, c_i^{syn})$
4:     ▷ ② Calculate reward for one sample:
5:     $r_i = \sum_{k=1}^m \log p_M(A_k^i \mid A_{<k}^i, T^l, \theta_M)$
6: end for
7: ▷ ③ Use the average as final reward:
8: $r = \frac{\sum_i r_i}{|\mathcal{D}_S^{syn}|}$
9: return $r$
```

For experiments on NQ dataset and TriviaQA dataset, we set B to 10 per gpu and we use 2 gpus to train, so B is actually 20. The learning rate of the selection agent is 3e-5, and we set $H_1 = H_3 = 768, H_2 = 64, C = 1000$ for both dataset. As for the number of training steps related to the convergence of RL training, we set the maximum number of training steps to 6000 and 5000 for the NQ dataset and the TriviaQA dataset, respectively, since the two datasets have different data volumes. However, we save checkpoints every 500 steps for judging convergence.

For experiments on FairytaleQA dataset, batch size B, learning rate and hidden layers dimension are the same as in the extractive datasets. The number of training steps varies with the amount of data. The scale of FairytaleQA dataset is not as large as extractive datasets, so we set the training step to 500 and save checkpoints every 100 steps for judging convergence.