Towards Better Question Generation in QA-based Event Extraction

Zijin Hong
Jinan University
hongzijin@stu2020.jnu.edu.cn

Jian Liu†
Beijing Jiaotong University
jianliu@bjtu.edu.cn

Abstract

Event Extraction (EE) is an essential information extraction task that aims to extract event-related information from unstructured texts. The paradigm of this task has shifted from conventional classification-based methods to more contemporary question-answering-based (QA-based) approaches. However, in QA-based EE, the quality of the questions dramatically affects the extraction accuracy, and how to generate high-quality questions for QA-based EE remains a challenge. In this work, to tackle this challenge, we suggest four criteria to evaluate the quality of a question and propose a reinforcement learning method, **RLQG**, for QA-based EE that can generate generalizable, high-quality, and context-dependent questions and provides clear guidance to QA models. The extensive experiments conducted on ACE and RAMS datasets have strongly validated our approach's effectiveness, which also demonstrates its robustness in scenarios with limited training data. The corresponding code of RLQG is released for further research[1].

1 Introduction

Event extraction (EE), an essential information extraction task, aims to extract event-related information (often called event arguments) from a given context. Recently, the paradigm of EE has shifted from conventional classification-based approaches (Li et al., 2013; Chen et al., 2015; Nguyen et al., 2016) to question-answering-based (QA-based) methods (Du and Cardie, 2020; Liu et al., 2020, 2021; Ma et al., 2022; Lu et al., 2023). For example, given a sentence: "*Marines were involved in a firefight in the center of Baghdad*", to extract the role *Attacker*, the method first generates the question "*Who is the attacker in firefight?*" and then uses a question-answering (QA) model to obtain an answer as the corresponding result (as shown in Figure 1). This paradigm has demonstrated great success in various scenarios/domains.

Context:
Marines$_{attacker}$ were involved in a firefight in the center of Baghdad. They used rocket-propelled grenades and semi-automatic weapons, causing some injuries.

Template Q1: Who is the attacker in firefight?
A1. None ✗

Template Q2: Who was the attacking agent?
A2. weapons ✗

Human-Written Q3: Who was involved in the firefight in the center of Baghdad using grenades and weapons?
A3. Marines ✓

Figure 1: An EE example from ACE dataset, with "firefight" being the event trigger and "Marines" being the event argument fulfilling the *Attacker* role.

Nevertheless, one of the biggest challenges in QA-based EE is obtaining "good questions" to guide the QA model, and the quality of the questions often significantly affects the results. Previous methods typically use well-designed templates to generate such questions (Du and Cardie, 2020; Zhou et al., 2023; Liu et al., 2023), which, however, often lead to rigid and less context-dependent questions. For example, Figure 1 gives two questions, Q1 and Q2, based on templates, leading to incorrect answers. In our pilot experiments, we show that template-based questions lead to about 60% errors even using a powerful proprietary QA model like GPT-4 (OpenAI, 2023), as shown in Figure 4.

In this paper, we explore effective methods towards generating better questions for QA-based EE. First, we propose four criteria for a good question: 1) Fluency: A question should be fluent in natural languages to be well addressed by a QA model. 2) Generalizability: Methods for question generation should apply to unseen contexts and roles beyond

† Corresponding author
[1] https://github.com/Rcrossmeister/RLQG

Findings of the Association for Computational Linguistics ACL 2024, pages 9025–9038
August 11-16, 2024 ©2024 Association for Computational Linguistics

those in training (Du and Cardie, 2020). 3) Context dependence: Questions should be consistent with the context and include necessary information to be correctly addressed by a QA model. 4) Indicative guidance for QA models: Questions should offer clear guidance for QA models to find answers (Kojima et al.). We then seek to build a model towards satisfying each aspect.

Methodologically, we develop a sequence-to-sequence-based text generation model that can learn from template questions, allowing it to generate more fluent questions and generalize to previously unexplored roles. Nonetheless, more is required to produce high-quality questions. Further, to meet the criteria of context dependence and indicative guidance, we developed a **R**einforcement **L**earning-based **Q**uestion **G**eneration framework, **RLQG** to refine the generation process (Christiano et al., 2017; Rafailov et al., 2023). Specifically, an inverse prompting mechanism is proposed to evaluate whether the question matches the context, and a question-answering reward is used to quantify the degree to which the question is indicative. We choose a positive and negative question pair based on the above two mechanisms and utilize these as signals to fine-tune the model, biasing it toward generating context-dependent and suggestive questions providing indicative guidance.

Finally, the effectiveness of our method has been verified on two widely used EE benchmarks. According to the results, on the full ACE (Doddington et al., 2004) and RAMS (Ebner et al., 2020) benchmarks, our method outperforms previous methods by 2.69% and 1.96%. More importantly, we show that our method is particularly effective in data-scarce scenarios – with only 40% of the training data, we achieved the same performance as previous works. Additionally, we show that we can achieve good performance based on simple questions without excessive manual intervention.

In summary, the contributions of our work are three-fold:

- We revisit question generation for QA-based EE and suggest four question evaluation criteria. We design a model that can generate better questions with these as guidance.

- We introduce a reinforcement learning framework for better question generation for EE, which is considered context-dependent and indicative of question generation.

- We have verified the effectiveness of our method on different benchmarks, and show its capability to handle the more challenging data-scarce scenario.

2 Related Work

2.1 QA-Based Event Extraction

Event extraction is an information extraction task focusing on extraction, particularly event information. Traditionally, methods formulate it as a classification problem (Ahn, 2006; Li et al., 2013; Chen et al., 2015; Nguyen et al., 2016), but recent methods have started a QA-based paradigm. The core is to generate a question to find each argument. For example, (Liu et al., 2020; Lyu et al., 2021) convert EE tasks into machine reading comprehension using simple questions that are highly generic. Then (Li et al., 2020) reformulate the task as multi-turn question answering, finish the trigger identification, and argument extraction by asking different questions. Recently, (Du and Cardie, 2020) and (Lu et al., 2023) studied the effect on question quality in question answering; they trained a question generation (QG) model to generate a better question and also fine-tuned a model to do the question answering to finish the EE task. Despite the above advances, there still exists a challenge regarding how to generate a “good” question and even what the definitions of “good questions” are. In this work, we provide four criteria for question generation and build a model towards satisfying each aspect.

2.2 Prompt Engineering for LLMs

Our work also relates to prompt engineering in LLMs. The prompt is the natural language input for the language model. Previous research has proven the efficiency of prompt engineering (Radford et al., 2019; Liu et al., 2023); practical prompt engineering can drastically improve the efficiency and output quality of language models, making them more useful across a wide range of applications, from creative writing and content generation (Zou et al., 2021) to technical problem-solving and data analysis (Chen et al., 2023). Recently, large language models (LLMs) became a main research object in language model study; as a chat model with general capacity in NLP task, prompt study began a crucial challenge in improving the performance of LLM’s response (Wei et al., 2022a). Studies like Chain of Thoughts (Wei et al., 2022b) and Retrieval

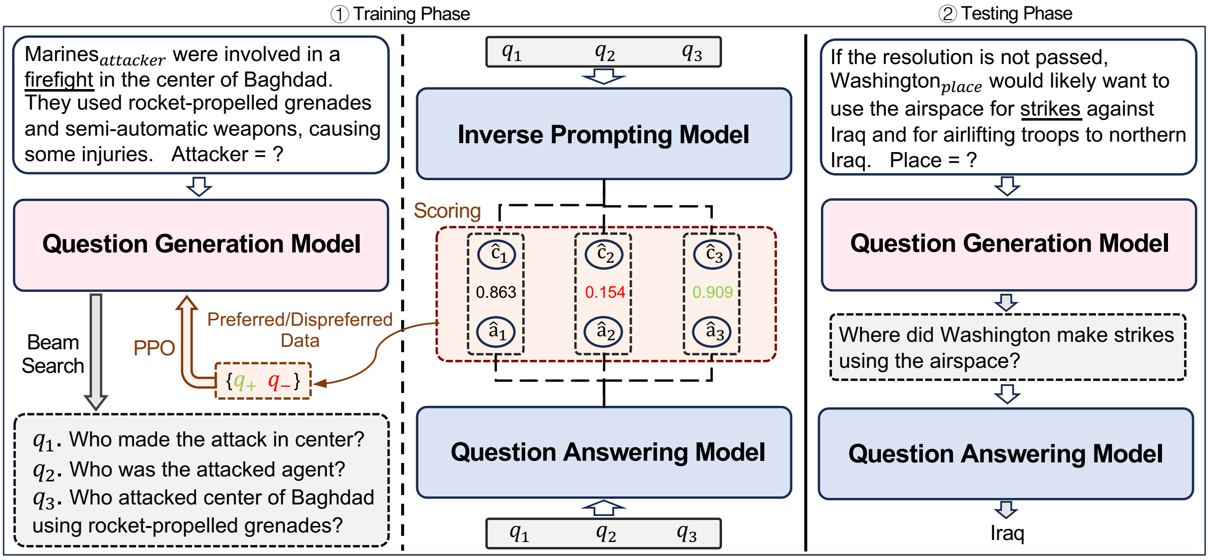

① Training Phase
② Testing Phase
Marines$_{attacker}$ were involved in a firefight in the center of Baghdad. They used rocket-propelled grenades and semi-automatic weapons, causing some injuries. Attacker = ?
Question Generation Model
Beam Search
PPO
Preferred/Dispreferred Data
$\{q_+ \ q_-\}$
q_1. Who made the attack in center?
q_2. Who was the attacked agent?
q_3. Who attacked center of Baghdad using rocket-propelled grenades?
q_1 q_2 q_3
Inverse Prompting Model
Scoring
$\hat{c}_1$ 0.863 $\hat{a}_1$
$\hat{c}_2$ 0.154 $\hat{a}_2$
$\hat{c}_3$ 0.909 $\hat{a}_3$
Question Answering Model
q_1 q_2 q_3
If the resolution is not passed, Washington$_{place}$ would likely want to use the airspace for strikes against Iraq and for airlifting troops to northern Iraq. Place = ?
Question Generation Model
Where did Washington make strikes using the airspace?
Question Answering Model
Iraq

Figure 2: The overview of our proposed method, comprises: 1) Training phase, including supervised fine-tuning and reinforcement learning refining for a question generation model. 2) Testing phase, a final (off-the-shelf) question-answering model predicts the final answer based on the given context and question.

Augmented Generation (Lewis et al., 2020) focus on involving information that helps the LLM for better contextual information for natural language reasoning and understanding. Our work is a particular case for generating prompts in EE to trigger LLM for information extraction.

3 Proposed Method

The overview of our method is visualized in Figure 2, which contains three main modules:

- A supervised fine-tuned (SFT) question generation model converts a given role and corresponding context into a question.
- A reinforcement learning (RL) framework refines the question generation through inverse prompting and question-answering rewards.
- A final (off-the-shelf) question-answering model to generate the final answer based on the given context and question.

We detail each part in the following.

3.1 Question Generation Module

The question generation module aims to generate a question for a semantic role. Assuming the context is c, annotated with an event trigger t, our goal is to generate a question q for the particular role r.

SFT over Template Questions. We utilize a general sequence-to-sequence framework as the backbone of the question generation model, and to make a good starting point, we use template questions as the targets for learning. Particularly, for a context, trigger, role triplet (c', t', r') in the training set, we construct the following input:

$$p_{(c',t',r')} = \text{role} : []_{r'} \oplus \text{trigger} : []_{t'} \oplus \text{context} : []_{c'} \quad (1)$$

Where $[]_{c'/t'/r'}$ denotes a slot to fill the context/trigger/role respectively, $\oplus$ is a concatenation operator. We then adopt the following template question[2] as the target:

$$q'_{(r',t')} = \text{WH}^{3} \text{ is the } []_{r'} \text{ in the } []_{t'} \text{ event?}$$

The question generation model learns a map from p to q' and is expressed as a probabilistic model. At testing time, the generated question q is formulated as follows:

$$q = \arg\max_{\tilde{q}} \Pr(\tilde{q} \mid p \,; \theta), \quad (2)$$

where θ denotes the parameter set of the model. In practice, we utilize LLaMA-2 (Touvron et al., 2023) as the base model, and the learning is performed by minimizing the cross-entropy loss func-

[2] In practice, we can apply more complex templates (Lu et al., 2023) as shown in Section 4.

[3] Interrogative pronoun.

Context: Warplanes$_{instrument}$ pounded forward Iraqi positions in the hills overlooking Chamchamal, 35 kilometers ...
Generated Question: What $instrument$ was used in the attack in Iraqi positions?
Recovered Context: An $instrument$ was used to pound the Iraqi positions during the attack.

Table 1: An example of inverse prompting mechanism from ACE. The recovered context is a basic rephrased description of the original context. More context-dependent questions will lead to higher similarity between recovered and original contexts.

tion $\mathcal{L}_c$ over each training instance:

$$\mathcal{L}_c = -\sum_{(c',t',r')\in D} \log \Pr(q'_{(r',t')} \mid p_{(c',t',r')}\ ;\theta), \tag{3}$$

where $\Pr(q \mid p)$ denotes the probability of generating q by given p, D is the training set.

Beam Search Augmentation. Considering that the standard question generation method can only get one question with the highest probability, to increase diversity, we adopt beam search augmentation to generate multiple sentences for further use. Remarkably, at each step of beam search, it keeps track of the N_{beam} most promising sequences, and N_{beam} is the beam size. And therefore, for each (c', t', r'), it can generate a set of candidate questions $\mathcal{Q} = \{q_1, q_2, ..., q_n\}$, n is the number of return questions given p.

3.2 RL for QG Refinement

To refine the questions, we build an RL framework with inverse prompting and question-answering reward. Our motivation is that if a question is context-dependent and indicative enough, we can use it to recover the context to some extent. When used as input to the QA model, it can yield the correct answer (during training). Then, we integrate the RL framework with the above two rewarding mechanisms. The overall training procedure is shown in Algorithm 1.

Inverse Prompting Reward. The inverse prompting mechanism aims to recover the context from the question. We assume that a better question is more context-dependent, and therefore, it is easier to recover the context to some extent. Here, we developed an inverse prompting model to achieve context recovery, which is a text generation model that takes the following inputs:

$$p^i_{(t',q)} = \text{trigger} : []_{t'} \oplus \text{question} : []_q \tag{4}$$

where q is the question previously generated by Eq. 2. The goal of inverse prompting is to recover the original context c; however, considering that it may contain information not appearing in the question, to ease the generation, we build a rephrased context $\hat{c}$ as a workaround. Particularly, for each role in the ACE ontology, we chose one example question and manually constructed $\hat{c}$ (given an example shown in Table 1), and we used ChatGPT to obtain more question-rephrased context pairs by using a few-shot prompting. Then, we train the inverse prompting model to recover $\hat{c}$ from p^i. More details are shown in Appendix A.

Finally, given a context c and its recovered context $\hat{c}$, we utilize semantic similarity (SemSim) to evaluate the quality of recovery, denoted as $\mathrm{SemSim}(c, \hat{c})$, which provides an inverse prompting reward for further use.

Question Answering Reward. Intuitively, a good question should successfully guide a QA model to yield a correct answer. Therefore, for each candidate question in $\mathcal{Q}$, we take it as the input of a QA model and generate a corresponding answer. For each question q with standard answer a, we use the context overlap ratio (COR) to evaluate the predicted answer $\hat{a}$, which is obtained by:

$$\mathrm{COR}(a, \hat{a}) = \frac{|a \cap \hat{a}|}{\max\{|a|, |\hat{a}|\}}, \tag{5}$$

the COR score is calculated at the word level, and we use the tokenizer from NLTK[4] for implementations. We then utilize $\mathrm{COR}(a, \hat{a})$ as the question-answering reward for further use.

Refining with RL. Typically, RL fine-tuning is employed subsequent to supervised fine-tuning for further refinement. We introduce a reinforcement learning framework to refine the QG module. Particularly, for each candidate question $q \in \mathcal{Q}$, we derive a combined score S_q according to inverse prompting and question answering reward:

$$S_q = \lambda_1 \, \mathrm{SemSim}(c, \hat{c}) + \lambda_2 \, \mathrm{COR}(a, \hat{a}), \tag{6}$$

[4] https://www.nltk.org/

Algorithm 1 RL for QG Refinement

```
for each $(c', t', r') \in$ training set $D$ do
    Generate question set $Q$ with $f_{SFT}$
    for each question $q \in Q$ do
        Generate recovered context $\hat{c}$
        Generate predicted answer $\hat{a}$
        Compute score using Eq. 6
    end for
    return Reward score set $S_Q$
    if Condition in Eq. 7 satisfied then
        return $(q_+, q_-)$ as Eq. 8
    end if
end for
Reward modeling $r(p, q)$ with Eq. 9
PPO training with objective function Eq. 10
return RL-refined model $f_{RL}$
```

the overall score of the question set Q can be represented as $S_Q = \{S_{q_1}, S_{q_2}, \ldots, S_{q_n}\}$. Next, we select preferred/dispreferred question pair from the question set Q according to the following criteria:

$$\begin{cases} \max(S_Q) > \alpha \\ \max(S_Q) - \min(S_Q) > \beta, \end{cases} \tag{7}$$

if the condition above is satisfied for Q, we return the question pair with the highest and lowest score:

$$(q_+, q_-) = (q_{\max(S_Q)}, q_{\min(S_Q)}), \tag{8}$$

and then combine the question pair with their corresponding input p to construct a preference dataset $D^{+/-}$. The reward modeling is minimizing the loss function for each $(p, q_+, q_-) \in D^{+/-}$:

$$\mathcal{L}_{\text{RM}} = -\mathbb{E}\left[\log\left(\sigma\left(r\left(p, q^+\right) - r\left(p, q^-\right)\right)\right)\right], \tag{9}$$

where σ is the sigmoid function, r is the score of question q given p. The purpose of reward modeling is to get a reward function $r(p, q)$, where the higher the reward $r(p, q)$, the better the question q is to the given input p. Denote the question generation model by supervised fine-tuned as f_{SFT}, the RL refining process is then to maximize the following objective function:

$$\mathcal{L}_{\text{RL}} = \mathbb{E}\left[r\left(p, q\right)\right] - \mu\mathbb{E}\,\text{KL}\left(f_{RL} \mid f_{SFT}\right), \tag{10}$$

where f_{RL} is the target model of the refinement, KL is Kullback-Leibler regularization, and $\mu > 0$ is the regularization parameter. This procedure enables a model to generate a better question with context-dependent and indicative refinement. Specifically, our RL framework utilizes the proximal policy optimization (PPO) algorithm (Schulman et al., 2017).

3.3 Question Answering Module

Finally, the event trigger, the target role, and the event context are given at the testing time. With an RL-refined model f_{RL}, we generate a question and use it as the prompt to trigger a QA model. Considering that the best performance models are usually proprietary models, we do not train an additional QA model like in previous works (Du and Cardie, 2020; Lu et al., 2023) but directly use an off-the-shelf QA model such as LLaMA-2-13b-Chat or ChatGPT. We enumerate each role and get the answer as the EE results.

4 Experimental Setups

4.1 Datasets

We conduct our experiments on the widely used dataset ACE 2005 (Doddington et al., 2004) and RAMS (Ebner et al., 2020). ACE 2005 has 33 event types and 22 argument roles, which contains 599 documents crawled between 2003 and 2005 from various areas. We follow the same data split and preprocessing step as in the prior works (Wadden et al., 2019). RAMS is a recently introduced dataset with document-level event argument extraction, which encompasses 9,124 annotated event mentions spanning 139 event types and 65 argument roles. More details of data preprocessing are given in Appendix B.

4.2 Evaluation Settings

In the ACE dataset, considering not all roles have related arguments in a given instance, we consider two settings for evaluations: (1) Practical evaluation: Only evaluate the questions whose target role has a corresponding argument (answerable questions). (2) Full evaluation: Evaluate the questions whose target role is all possible roles in ACE ontology (some of them are unanswerable questions) (Rajpurkar et al., 2018).

In the RAMS dataset, all the questions are in practical evaluation. For evaluation metrics, considering that exactly matching the QA model's response with the standard answer is difficult for an off-the-shelf model, we selected several metrics of varying degrees.

The metric considers: 1) Exact Match Accuracy (EM), a predicted answer is considered correct only if all of it exactly matches the standard answer. 2) Context Overlap Ratio (COR), which can be computed with Eq. 5. 3) Semantic Similarity (SemSim), which can evaluate the detailed response.

4.3 Implementations

In our implementations of the QG module, we use LLaMA-2-7b (Touvron et al., 2023) as the base model; the learning rate is set as 5e-5, selected from the interval [1e-5, 1e-4]. For the QA model, we adopt the freeze (off-the-shelf) model in both training and testing, LLaMA-2-13b-Chat, and we also adopt a 5-shot in prompting to help the QA model's understanding. All the question-answering processes use the same few-shot examples and this setting. The selection of preferred/dispreferred pairs is selected according to condition Eq. 7 and reward score Eq. 6. In practice, to balance the model's performance, we set λ_1 to 0.3 and λ_2 to 0.7 in Eq. 6. In Eq. 7, the α is set to 0.65, and β is set to 0.5. The discussion of model selection and the details of training and hyperparameters are given in Appendix C.

4.4 Baselines

We divide baselines into three groups: 1) Template-based methods: RCEE (Liu et al., 2020), which uses a simple template such as "WH is the [role]?", denoted as Simple-Q[5]. EEQA (Du and Cardie, 2020), which introduces two types of questions: a template that incorporates the trigger and Simple-Q, denoted as Standard-Q. Moreover, the question generated by the descriptions of each argument role is provided in the ACE annotation guidelines for events, denoted as Guideline-Q. Back-Translation, which back-translate the Standard-Q denoted as Back-Trans-Q. QGA-EE (Lu et al., 2023), which designed a dynamic template for different event type and their corresponding role, denoted as Dynamic-Q. 2) Supervised fine-tuning-based methods: Which trains a QG model via supervised fine-tuning using the above templates[6]. They are denoted as SFT (*Template*) respectively. 3) In-context learning-based methods: These perform 0-shot and 5-shot on LLaMA-2-13b-Chat and GPT-4, respectively. The few-shot example will be shown in Appendix D. Noting that the above methods only studied and designed the template questions on ACE, considering the generalizability of the method above, we adopt Standard-Q and Back-Trans-Q as the RAMS experimental baselines. Our proposed method, **R**einforcement **L**earning-based **Q**uestion **G**eneration refinement, is called **RLQG**. Specifically, we train our model on the ACE dataset based on the most contextualized template, Dynamic-Q. On the RAMS dataset, we train the model based on a more natural question, Back-Trans-Q. We will discuss the influence of different template starting points in Section 6.1.

5 Experimental Results

In this section, we present the experimental results divided by results with full training resources on ACE and RAMS and results on the data-scarce scenarios in ACE.

5.1 Results with Full Training Resource

Table 2 gives the results on ACE with full training resources. From the results: 1) Above all baseline methods we compared, it is evident that our method surpasses all the baselines in terms of three given metrics and evaluation settings. Specifically, our method outperforms the second-best method SFT (*Dynamic*) by 2.08% and outperforms the template starting point Dynamic-Q by 2.69% in EM, with practical evaluation. We attribute the results to the RL refinement with rewards of inverse prompting and question answering in our framework, which helps our question become more context-dependent and indicative than the other method. We also surpassed the GPT-4 (*5shot*) in evaluation, demonstrating the powerful potential of our model compared to the proprietary model. 2) By comparing different baselines, we explore that the quality of template questions determines the ability of the SFT model for question generation. Also, a higher quality template as the starting point to train the SFT model will improve the SFT process. The results with in-context learning methods show that the model's ability affects the performance, and the few-shot example also brings a guideline on better question generation.

Table 3 gives the evaluation results of the RAMS test set. From the results: 1) It is also evident that our method RLQG outperforms the comparative baseline on all metrics, which obtains 1.32% improvement on EM compared to the second-best

[5] Q stands for Question.

[6] Using different templates in Section 3.1.

Methods	Practical Eval.			Full Eval.		
	EM	COR	SemSim	EM	COR	SemSim
Template						
Simple-Q (Liu et al., 2020)	35.41	40.23	60.93	14.38	16.17	24.55
Standard-Q (Du and Cardie, 2020)	37.42	43.87	63.70	15.60	17.36	25.92
Back-Trans-Q	36.13	41.39	62.41	15.14	16.67	25.28
Guideline-Q (Du and Cardie, 2020)	38.51	45.28	65.54	17.61	19.96	28.14
Dynamic-Q (Lu et al., 2023)	38.70	45.79	65.55	20.45	23.12	30.79
Supervised Fine-tuning						
SFT (*Standard*)	37.63	42.95	62.36	15.31	17.13	25.72
SFT (*Back-Trans*)	38.24	43.56	64.11	17.47	18.90	27.32
SFT (*Guideline*)	38.62	44.69	64.66	17.33	19.61	27.77
SFT (*Dynamic*)	39.31	46.78	66.24	20.35	23.05	30.53
In-context learning						
LLaMA-2-13b-Chat (*0shot*)	1.21	3.50	35.88	0.43	1.25	21.78
LLaMA-2-13b-Chat (*5shot*)	27.97	33.04	53.69	13.01	14.93	23.54
GPT-4 (*0shot*)	28.97	35.83	57.90	11.14	13.54	23.35
GPT-4 (*5shot*)	39.24	47.59	65.92	16.32	19.37	27.46
RLQG (Ours)	**41.39**	**48.58**	**67.94**	**21.71**	**24.19**	**31.80**

Table 2: Event extraction results with Practical Evaluation and Full Evaluation on the ACE test dataset, where EM, COR, and SemSim indicate exact match accuracy, context overlap ratio, and semantic similarity, respectively.

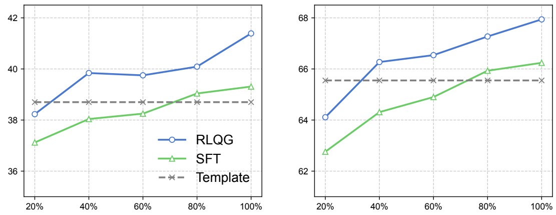
Methods	EM	COR	SemSim
Template			
Standard-Q	17.65	23.02	47.96
Back-Trans-Q	16.45	21.47	46.43
Supervised Fine-tuning			
SFT (*Standard*)	18.10	23.84	48.79
SFT (*Back-Trans*)	18.29	24.11	49.32
RLQG (Ours)	**19.61**	**25.43**	**50.69**

Table 3: Event extraction results on the RAMS test dataset with practical evaluation.

model SFT (*Back-Trans*), and also outperforms the best template method Standard-Q by 1.96%. 2) Compared to the static form Standard-Q, the improvement of fine-tuning to the Back-Trans-Q is more significant; we conclude that the model improves more on natural questions.

5.2 Results in Data-Scarce Scenarios

To explore the performance of our model in the data-scarcity settings, we randomly choose x% number of dynamic templates (Lu et al., 2023) to fill in and use these questions as our training data.



(a) Exact Match Accuracy (b) Semantic Similarity

Figure 3: Experimental results in the ACE dataset for the data-scarce scenario. The x-axis represents different ratios of training data, y-axis is the value of the metric.

The comparative baseline is the template-based method Dynamic-Q and supervised fine-tuning-based method SFT (*Dynamic*). The results shown in Figure 3a and 3b represent their performance on the different metrics in the data-scarce scenarios.

From the results, it is significant that our method outperforms the SFT method in data scarcity, especially when the data is limited to around 40% to 60%; our method opens up a big gap with the fine-tuning method. In addition, we can outperform the full-annotated template by using only 40% of training data, performing a good adaptation with a data-scarce scenario and budget limitations. As further explanation, the SFT method can only learn from

Methods	ACE	RAMS
Standard-Q	37.42	17.65
SFT (*Standard*)	37.63	18.10
RLQG (*Standard*)	**39.29**	**19.23**

Table 4: Ablation study of the different template as the starting point on exact match accuracy

Method	EM	COR	SemSim
RLQG	**41.39**	**48.58**	**67.94**
-w/o *IP reward*	40.21	47.49	66.96
-w/o *QA reward*	39.86	47.17	66.88

Table 5: Results of ablation studies for removing different reward.

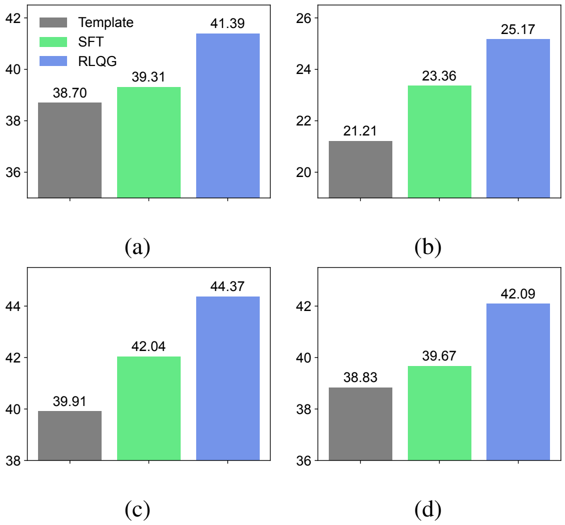


Figure 4: The performance with exact match accuracy on different QA model (a) LLaMA-2-13b-Chat (*5shot*) (identical to previous settings) (b) LLaMA-2-13b-Chat (*0shot*) (c) GPT-4 (*5shot*) (d) GPT-4 (*0shot*)

annotated data, and when training data is limited, it struggles with generalization issues, resulting in poor performance. In contrast, our method is more data-efficient because it combines two mechanisms: 1) The QG model, combined with the beam search mechanism, can generate more diverse questions, potentially providing more supervision signals. 2) More importantly, our RL refinement mechanism assists in ranking such questions and identifying better ones for optimization, resulting in improved generalization ability. By combining the two mechanisms, our method can make the best use of limited data while producing superior results.

6 Further Discussion

If not explicitly stated, further discussion of our method's details focuses on the representative baseline with the template-based method Dynamic-Q and its corresponding SFT model. The experiments are conducted on the ACE dataset with practical evaluation.

6.1 Ablations on QG Architecture

Template Starting Point. We turn our method's starting point on the template question that with the most minor human intervention: Standard-Q, which can obtained by simply concatenating the interrogative pronoun, the role, and the trigger word. The experiments are conducted in both datasets on the metric EM. Table 4 shows the corresponding results, in which our method outperforms the template-based and supervised fine-tuning-based method in both the ACE and RAMS datasets. This is a competitive result that our method can also achieve good performance even if there is only rarely manual intervention.

Rewards of Two Mechanism. We conduct the ablation study on inverse prompting (IP) and question answering (QA) rewards; Table 5 lists different variations of our proposed method. When removing the IP reward, the exact match accuracy (EM) decreased by 1.18%, and the COR and SemSim decreased by 1.09% and 0.98%, respectively. And without QA reward, the decrease became 1.53%, 1.41%, and 1.06% to these three metric. These results indicate that the indicative is prior in question generation. Overall, each component of the RLQG method plays a crucial role in achieving good performance, as their removal resulted in decreased performance across all metrics.

6.2 Ablation on QA architectures

This section discusses the impact of different QA models in our method, we explore the universal evaluation of the question quality by adopting different QA modules. As a prompting aspect, we compare the QA module with 5-shot and 0-shot. Moreover, considering the model aspect, we utilize GPT-4 as a comparative QA model. As shown in Figure 4, our method outperforms the other two methods with different QA models in all cases. The basic capacity of the QA model will lead to the diversity of the performance on EAE. However, our method represents suitability for different selections of QA models.

Context: Former senior banker Callum McCarthy$_{person}$ **begins** what is one of the most important jobs in London's financial world in September, when incumbent Howard Davies steps down.
Template Question: Who is the employee? **Answer:** Davies ✗
SFT Question: Who was hired by banker? **Answer:** Howard Davies ✗
RLQG Question: Who was hired as one of the most important jobs? **Answer:** Callum McCarthy ✓
Human-Written Question: Who was the former senior banker that began an important job in September? **Answer:** Callum McCarthy ✓

Table 6: An instance of the ACE test set, we use different methods to generate a corresponding question on semantic role *person*; our method gets the correct answer and gets close to a human-written question.

6.3 Case Study

As expected, our method will be a better question regarding fluency, context dependence, and generalizability. We present an intuitive analysis of a case study. As shown in Table 6, we select an example in the ACE05 test dataset. The target of these questions is to trigger the QA model to extract the answer *McCarthy*. Obviously, the question generated by RLQG get the correct answer and is most similar to human-written questions.

7 Conclusion

Event extraction (EE) has evolved from traditional classification into question-answering-based methods (QA-based EE). These methods emphasize the design of quality questions to guide QA models for better answers. In this paper, we introduce a reinforcement learning framework that aims to produce context-dependent, fluently phrased questions that are generalizable and indicative enough to QA models, addressing the challenges posed by rigid, template-based questions. Our methodology demonstrates improved performance on ACE and RAMS benchmarks, particularly in data-scarce scenarios. It highlights the method's efficacy in generating effective questions for QA-based EE without extensive manual intervention.

8 Limitations

Two primary limitations are acknowledged in this study. Firstly, most existing QA-based EE approaches assume known triggers, effectively overlooking the impact of trigger identification. This study follows this assumption but plans to incorporate event detection (ED) in future work for a comprehensive approach to the event extraction (EE) task. Secondly, the method's generalizability to real-world scenarios remains to be determined, as it has only been evaluated on standard datasets. The complexity, diversity, and potential noise of real-world data call for further validation to confirm the method's effectiveness in practical applications. Additionally, our research raises no ethical issues because it focuses solely on the technical aspects of a normal information extraction problem.

References

David Ahn. 2006. The stages of event extraction. In *ACL 2006 Workshop on Annotating and Reasoning about Time and Events (ACL)*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research (TMLR)*.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 2024. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335*.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems (NeurIPS)*.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph

Weischedel. 2004. The automatic content extraction (ACE) program – tasks, data, and evaluation. In *International Conference on Language Resources and Evaluation (LREC)*.

Xinya Du and Claire Cardie. 2020. Event extraction by answering (almost) natural questions. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Seth Ebner, Patrick Xia, Ryan Culkin, Kyle Rawlins, and Benjamin Van Durme. 2020. Multi-sentence argument linking. In *Association for Computational Linguistics (ACL)*.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Fayuan Li, Weihua Peng, Yuguang Chen, Quan Wang, Lu Pan, Yajuan Lyu, and Yong Zhu. 2020. Event extraction as multi-turn question answering. In *Findings of Empirical Methods in Natural Language Processing (EMNLP)*.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Association for Computational Linguistics (ACL)*.

Sha Li, Heng Ji, and Jiawei Han. 2021. Document-level event argument extraction by conditional generation. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

Jian Liu, Yubo Chen, Kang Liu, Wei Bi, and Xiaojiang Liu. 2020. Event extraction as machine reading comprehension. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Jian Liu, Yufeng Chen, and Jinan Xu. 2021. Machine reading comprehension as data augmentation: A case study on implicit event argument extraction. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*.

Di Lu, Shihao Ran, Joel Tetreault, and Alejandro Jaimes. 2023. Event extraction as question generation and answering. In *Association for Computational Linguistics (ACL)*.

Qing Lyu, Hongming Zhang, Elior Sulem, and Dan Roth. 2021. Zero-shot event extraction via transfer learning: Challenges and insights. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

Yubo Ma, Zehao Wang, Yixin Cao, Mukai Li, Meiqi Chen, Kun Wang, and Jing Shao. 2022. Prompt for extraction? PAIE: Prompting argument interaction for event argument extraction. In *Association for Computational Linguistics (ACL)*.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research (JMLR)*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Association for Computational Linguistics (ACL)*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Tianyi Tang, Junyi Li, Wayne Xin Zhao, and Ji-Rong Wen. 2022. Context-tuning: Learning contextualized prompts for natural language generation. In *International Conference on Computational Linguistics (COLING)*.

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*.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. In *Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022a. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations (ICLR)*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022b. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Shaochen Zhong, Bing Yin, and Xia Hu. 2024. Harnessing the power of llms in practice: A survey on chatgpt and beyond. *ACM Transactions on Knowledge Discovery from Data*.

Yuqing Yang, Qipeng Guo, Xiangkun Hu, Yue Zhang, Xipeng Qiu, and Zheng Zhang. 2023. An AMR-based link prediction approach for document-level event argument extraction. In *Association for Computational Linguistics (ACL)*.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, Weng Lam Tam, Zixuan Ma, Yufei Xue, Jidong Zhai, Wenguang Chen, Zhiyuan Liu, Peng Zhang, Yuxiao Dong, and Jie Tang. 2023. GLM-130b: An open bilingual pre-trained model. In *International Conference on Learning Representations (ICLR)*.

Qi Zeng, Qiusi Zhan, and Heng Ji. 2022. EA^{2}E: Improving consistency with event awareness for document-level argument extraction. In *Findings of North American Chapter of the Association for Computational Linguistics (NAACL)*.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyan Luo. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. *arXiv preprint arXiv:2403.13372*.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023. Large language models are human-level prompt engineers. In *International Conference on Learning Representations (ICLR)*.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. 2023. A survey on model compression for large language models. *arXiv preprint arXiv:2308.07633*.

Xu Zou, Da Yin, Qingyang Zhong, Hongxia Yang, Zhilin Yang, and Jie Tang. 2021. Controllable generation from pre-trained language models via inverse prompting. In *International Conference on Knowledge Discovery and Data Mining (KDD)*.

A Details of Inverse Prompting

A.1 Few-shot Instance of Inverse Prompting

We provide a few-shot learning instance of manually labeled recovered context in Table 7. We train the inverse prompting model using 97 question-rephrased context pairs generated by ChatGPT.

System: You are a helpful assistant. I'm giving you a question and an event trigger, please use them to recover the context of the event. Try your best to include as much information as possible.
User: trigger: bankruptcy question: What organization will declare bankruptcy soon? **Assistant:** An organization is soon to declare bankruptcy.
User: trigger: bankruptcy question: Where did WorldCom declare the bankruptcy? **Assistant:** WorldCom declared bankruptcy in somewhere.
User: trigger: fall question: What organization was ended by iraqis? **Assistant:** An organization was ended by Iraqis during a fall.
User: trigger: fallen question: Where did dictator Suharto fallen and democratic elections executed? **Assistant:** Dicatator Suharto was fallen and democratic elections were executed somewhere.
User: trigger: founded question: Who started the automaker in 1937? **Assistant:** Someone founded the automaker in 1937

Table 7: Few-shot learning instance of inverse prompting.

A.2 Variations of Inverse Prompting Model

We utilize the supervised fine-tuning inverse prompting model in our setting. Moreover, we also discuss the performance with some variations of the inverse prompting model (IPM): 1) Removing the SFT process that directly uses an off-the-shelf model instead (specifically, LLaMA-2-7b-Chat). 2) Replacing the SFT process with 5-shot learning. The few-shot learning instances are collected from Table 7.

Method	EM	COR	SemSim
RLQG	41.39	48.58	67.94
-1) w/o SFT IPM	40.35	47.37	66.12
-2) w/ 5-shot IPM	41.01	48.23	67.42

Table 8: Performance with Inverse Prompting Model Variations.

Table 8 shows the corresponding results. Notice that directly recovering the context is challenging, but using few-shot examples obtains a solid performance without fine-tuning.

B Data Preprocessing

B.1 ACE2005

We follow the step to preprocess and split the data in (Wadden et al., 2019). To get the template question, the simple template (Du and Cardie, 2020; Liu et al., 2020) can be directly adopted, and the dynamic template (Lu et al., 2023) needs to be filled in. We followed their proposed method to obtain the questions.

B.2 RAMS

The dataset is officially split and in natural language type in https://nlp.jhu.edu/rams. We get the question by simply combining the opponent of trigger and role as equation 1. The back translation question is obtained by Google document translation[7] by translating the question to Chinese and back to English.

C Implementation Details

We used parameter-efficient fine-tuning (PEFT) in the previous training stage to train our models. Specifically, in each stage (supervised fine-tuning and reinforcement learning), we utilize low-rank adaptation (LoRA) (Hu et al., 2021) as PEFT method, the trainable parameters occupy 0.0622% of full parameters. Every random seed in our experiments is set to 42. The details of training and hyper-parameters are as follows.

C.1 Supervised Fine-tuning

As previously introduced, the base model of question generation we selected is LLaMA-2-7b; the

[7] https://translate.google.com

training details are listed in Table 9. The model's architecture is identical to the official provided in Huggingface. Also, we train the inverse prompting model with identical setting.

Hyper-parameters	Value
data type	bf16
learning rate	5e-05
number of epochs	3
number of batch size	16
gradient accumulation steps	4

Table 9: Hyper-parameters of the SFT stage of question generation model's training

We tested our method's performance on different base model selections, including ChatGLM (Zeng et al., 2023) and Qwen (Bai et al., 2023). The results are shown in Table 10; our method also gets the best performance.

Models	**EM**	**COR**	**SemSim**
ChatGLM-3-6b(SFT)	38.63	45.09	64.67
ChatGLM-3-6b(RLQG)	40.45	46.97	66.16
Qwen-7b(SFT)	41.45	48.48	67.43
Qwen-7b(RLQG)	43.22	50.04	69.08

Table 10: Performance on the ACE test set with different base model selections.

We also verified our method on different model's size and variations, as shown in Table 11.

Model Size	**EM**	**COR**	**SemSim**
LLaMA-2-7b	41.39	48.58	67.94
LLaMA-2-7b-Chat	41.36	49.01	67.85
LLaMA-2-13b	41.53	48.77	67.81
LLaMA-2-13b-Chat	41.49	48.68	67.83

Table 11: Performance on the ACE test set with different model sizes and variations.

C.2 PPO Training

The hyper-parameters of reinforcement learning stage is similar with the previous stage, the details are shown in Table 12.

Notice that the learning rate of reward model training is set to 1e-6.

Hyper-parameters	Value
data Type	bf16
learning rate	1e-05
number of epochs	1
number of batch size	8
gradient accumulation steps	4

Table 12: Hyper-parameters of the RL stage of question generation model's training

C.3 Generation Configuration

In each generation part of our framework, including the training stage, question prediction, and question answering, the configurations are identical. The details are listed in Table 13.

Configuration	Value
top p	0.9
do sample	True
temperature	0.6
max token length	4096
predict with generate	True

Table 13: Generation configuration

Notice that, in beam search augmentation in Section 3.1, the configurations are slightly different; the "do sample" option should be "True", and the number of beams is set to 10, and the number of return sentences is 5.

C.4 Versions of Proprietary LLMs

The version of ChatGPT in this work is GPT-3.5-Turbo-1106 (OpenAI, 2023), and version of GPT-4 is GPT-4-1106-preview (OpenAI, 2023).

D Few-shot Learning Details

In few-shot learning, the examples are directly combined with the input prompt by setting different characters in front of the context.

D.1 Question Generation with Few-shot Learning

In section 4, we compared our method with method LLaMA-2-13b-Chat *(5-shot)* and GPT-4 *(5-shot)*. The prompt details are described in Table 14.

D.2 Question Answering with Few-shot Learning

The question answering in our research are conducted under calling a freeze chat model: LLaMA-2-13b-Chat with 5-shot. In the ablation study that using different models for question answering are still using the same prompt with identical examples. Details are listed in Table 15

System:
You are a helpful assistant. Please generate a natural language question with contextual information. The question aims to help language models to extract roles from context. Generate a question based on the even trigger given.

User:
role: agent trigger: election context: He lost an * election * to a dead man.
Assistant: Who was the voting agent?

User:
role: person trigger: quit context: Media tycoon Barry Diller on Wednesday * quit * as chief of Vivendi Universal Entertainment, the entertainment unit of French giant Vivendi Universal whose future appears up for grabs.
Assistant: Who was quit as chief of Vivendi Universal Entertainment?

User:
role: vehicle trigger: landed context: Even as the secretary of homeland security was putting his people on high alert last month, a 30-foot Cuban patrol boat with four heavily armed men * landed * on American shores, utterly undetected by the Coast Guard Secretary Ridge now leads.
Assistant: What vehicle was used for transporting men to shores?

User:
role: entity trigger: Former context: NOVAK * Former * Arkansas Governor and U.S. Senator Dale Bumpers has just published a memoir called "The Best Lawyer in a One-Lawyer Town." And it spans his life from the depression era to the Clinton era.
Assistant: Who fired Dale Bumpers?

User:
role: place trigger: war context: It could swell to as much as $500 billion if we go to * war * in Iraq.
Assistant: Where did the war take place?

Table 14: Few-shot learning instance of question generation

System:
You are a precise and concise assistant. Your task is to extract some words based directly on the provided context to answer the given questions. Please wrap your answer with the following tags: [ANS] [/ANS]. If a question has multiple correct answers within the context, list them all, separated by commas. If there is no answer in the context, just reply [ANS] None [/ANS]. Do NOT add any introductory phrases, explanations, or additional information outside of the given context.

User:
question: Who made the battle in Baghdad? context: US Secretary of Defense Donald Rumsfeld dismissed worries that there were insufficient forces in the Gulf region if the battle for Baghdad goes wrong.
Assistant: [ANS] US [/ANS]

User:
question: Who was nominated? context: Senator Christopher Dodd of Connecticut made the announcement today that he would not be the 10th candidate for the nomination.
Assistant: [ANS] candidate [/ANS]

User:
question: Who is person in former event? context: We're talking about possibilities of full scale war with former Congressman Tom Andrews, Democrat of Maine.
Assistant: [ANS] candidate [/ANS]

User:
question: Who died that cause Clinton suffered greatly? context: Clinton suffered greatly over the 19 Rangers that died, 18 on the 3rd of October and Matt Reersen (ph) three days later.
Assistant: [ANS] Rangers, Matt Reersen [/ANS]

User:
question: Where did the election takes place? context: He lost an election to a dead man.
Assistant: [ANS] None [/ANS]

Table 15: Few-shot learning instance of question answering