TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 04/14 pp464–474
DOI: 10.26599/TST.2022.9010034
Volume 28, Number 3, June 2023

Leveraging Structured Information from a Passage to Generate Questions

Jian Xu, Yu Sun*, Jianhou Gan, Mingtao Zhou, and Di Wu

Abstract: Question Generation (QG) is the task of utilizing Artificial Intelligence (AI) technology to generate questions that can be answered by a span of text within a given passage. Existing research on QG in the educational field struggles with two challenges: the mainstream QG models based on seq-to-seq fail to utilize the structured information from the passage; the other is the lack of specialized educational QG datasets. To address the challenges, a specialized QG dataset, reading comprehension dataset from examinations for QG (named RACE4QG), is reconstructed by applying a new answer tagging approach and a data-filtering strategy to the RACE dataset. Further, an end-to-end QG model, which can exploit the intra- and inter-sentence information to generate better questions, is proposed. In our model, the encoder utilizes a Gated Recurrent Units (GRU) network, which takes the concatenation of word embedding, answer tagging, and Graph Attention neTworks(GAT) embedding as input. The hidden states of the GRU are operated with a gated self-attention to obtain the final passage-answer representation, which will be fed to the decoder. Results show that our model outperforms baselines on automatic metrics and human evaluation. Consequently, the model improves the baseline by 0.44, 1.32, and 1.34 on BLEU-4, ROUGE-L, and METEOR metrics, respectively, indicating the effectivity and reliability of our model. Its gap with human expectations also reflects the research potential.

Key words: automatic Question Generation (QG); RACE4QG dataset; Answer-Oriented GAT (AO-GAT); attention mechanism; structured information

1 Introduction

Reading is a vital skill of human communication in the digital age[1]. One of the most important teaching objectives in non-English speaking countries is to improve learners' reading comprehension[2]. Unfortunately, in practice, achieving the teaching goal is not always ideal. The main reasons are due to outdated and inflexible teaching materials (i.e., passages and comprehension questions). New materials help keep instruction current and fresh, and allow teachers to tailor lessons to the interests of particular student populations. This necessitates that teachers generate comprehension questions quickly whenever they incorporate new text into the curriculum. However, it is difficult to write diverse and attactive comprehension questions. If teachers can generate comprehension questions efficiently and quickly, the teaching quality of English courses will be significantly enhanced, as will students' reading comprehension skills.

Fortunately, Artificial Intelligence (AI) has grown by

• Jian Xu and Di Wu are with the Key Laboratory of Educational Informatization for Nationalities, Yunnan Normal University, Kunming 650500, China. Jian Xu is also with the School of Information Engineering, Qujing Normal University, Qujing 655011, China. E-mail: qjncxj@126.com; skaldjfhg@qq.com.
• Yu Sun and Mingtao Zhou are with the School of Information Science and Technology, Yunnan Normal University, Kunming 650500, China. E-mail: sunyu@ynnu.edu.cn; zmtjavazmt@163.com.
• Jianhou Gan is with the Yunnan Key Laboratory of Smart Education, Yunnan Normal University, Kunming 650500, China. E-mail: ganjh@ynnu.edu.cn.
∗ To whom correspondence should be addressed.
Manuscript received: 2022-07-04; revised: 2022-08-10; accepted: 2022-08-22

leaps and bounds in recent years, shedding light on the issue. Automatic Question Generation(QG) is an NLP task and a significant branch of AI. QG task, which aims at generating fluent and correct questions on the basis of a given context and answers, has attracted many researchers in recent years from the natural language processing community[3–6]. The state-of-the-art models commonly utilize the neural QG strategy, where a neural network is trained on the basis of a seq-to-seq (i.e., sequence-to-sequence) backbone. Due to the complexity of QG, the best QG model only achieves the BLEU-4 score of 11.19 in the educational field[7].

In essence, the QG task aims to train a neural model that can automatically generate questions on the basis of a dataset. Therefore, the performance of the QG model depends primarily on its dataset and model. Although there exist some related datasets, such as the SQuAD[8], Narrative QA[9], Hotpotqa[10], and Race[11], they cannot be directly used for our QG task, for reasons in the following: (1) most of these datasets are general-purpose datasets rather than educational domain-specific datasets, and these datasets are Question Answering (QA) datasets where each sample is a quadruple (passage, question, answer, and distractor), whereas each sample of a QG dataset is a triple (passage, answer, and question). (2) The existing QG models mainly use Recurrent Neural Networks (RNN), such as the Long Short-Term Memory Network (LSTM)[12]; however, RNN models have an inherent sequential nature and find it difficult to handle long input sequences, forcing these RNN models to generate questions with only sentence-level information rather than passage-level information[3, 6]. To address the challenges, Du et al.[3] proposed to utilize sentence-level information to improve the performance of QG models. Given that Du et al.[3] cannot perform well for passage-level input (i.e., multiple sentences), Zhao et al.[5] introduced a maxout-pointer and gated self-attention mechanism, and achieved the best performance. However, even the best QG models are dysfunctional when generating comprehension questions for the educational domain. The dysfunction can be attributed to a double challenge: the quality of the education-type QG dataset and the performance of the QG model. In education, addressing these issues is crucial, as teachers can use automatically generated questions to increase classroom engagement and assess reading ability.

This paper investigates the RACE, an educational QA dataset, and finds that it can be adjusted for our QG task by removing irrelevant data and performing a new answer tagging. Moreover, considering the poor performance of the existing QG models, we propose a more powerful QG model with a more advanced encoder and decoder.

The contributions of this paper can be summarized as follows:

(1) Reconstruct a QG dataset, reading comprehension dataset from examination for QG (named RACE4QG), on the basis of the existing educational dataset RACE. We provide a new answer tagging strategy with similarity, and this strategy can inject richer additional information into RACE4QG.

(2) On the basis of the RACE4QG dataset, an end-to-end framework for QG is proposed, and the framework mainly comprises an encoder and a decoder. In the encoder, a Graph Attention neTworks (GAT) mechanism is applied to enrich the word embedding. In the decoder, an attention mechanism and pointer network are used to generate questions dynamically.

(3) Conduct several comparative and ablation experiments. In addition, case studies and human evaluations are conducted.

The rest of this paper is organized as follows. Section 2 discusses the related work for question generation. Section 3 focuses on building the end-to-end framework for generating questions, using the Answer-Oriented GAT (AO-GAT) strategy for the encoder and a pointer network mechanism for the decoder. Section 4 presents the multiple conducted experiments for our model. Section 5 provides the results and analysis. Section 6 concludes the paper and points out future research directions.

2 Related Work

Generating questions for reading practice and assessment is the foremost application of QG, which has been studied for many years[4, 13, 14].

Traditionally, QG is primarily based on heuristic rules that use manually built templates to generate questions and rank the generated questions[15–17]. Heilman and Smith[15] first proposed to generate questions through a heuristic strategy, which utilizes manually written rules to conduct syntactic transformations that turn declarative sentences into questions; then, they ranked the generated questions by a logistic regression model to select the best questions. Yao et al.[17] proposed a semantic rewriting approach (i.e., minimal recursion semantic representation) to build semantic structures

and grammar rules to generate better questions. In view of many challenges of creating full semantic representation, Labutov et al.[16] sought to bypass the challenges, they streamlined the QG task into an ontology-crowd-relevance workflow, then crowdsourced questions templates, and finally selected the best candidate templates to generate questions. These QG methods are mainly based on the heuristic rules, which restrict the generalization ability of models in various fields.

However, building well-structured rules is time-consuming and labor-intensive. Since the mid-2010s, there has been a greater interest in employing statistical methods, especially neural networks[3, 5, 6].

Currently, seq-to-seq methods with attention mechanisms have become mainstream, aiming to generate questions by training a neural network using a seq-to-seq framework[3, 5, 6, 18, 19]. Du et al.[3] first used a seq-to-seq neural model to generate questions and revealed that the seq-to-seq model achieves great performance improvement beyond the previous rule-based models. However, they also admitted that the generated questions cannot accurately correspond to some parts of the original context. To address this challenge, Zhou et al.[6] proposed to incorporate the position information of answer words (words occurring in the answer text) into the encoding process using an annotation vector. Rather than employing the annotation vector to tag answer positions, Song et al.[18] proposed to use a unified framework to encode both the passage and the answer. However, the above work is dysfunctional when dealing with a long input context. Zhao et al.[5] introduced a mixed mechanism that uses a gated self-attention and maxout pointer to process the long input context. Following the work of Zhao et al.[5], Yuan et al.[19] first obtained deep linguistic features by training large pretrained neural models on some NLP tasks and then incorporated these deep linguistic features into a seq-to-seq QG model to guide the question generation. The QG model improved the baseline by 6.2% on the BLEU-4 metric.

The aforementioned models are generic and not specific to the field of education. Moreover, they fail to effectively utilize the structural information within passages.

3 Our Model

3.1 Model overview

We propose an end-to-end framework for the automatic QG on the basis of a reconstructed dataset, then generate many grammatically consistent and fluent questions according to the given input passage and answer.

Formally, we use p, a, and q to represent the passage, answers, and questions, respectively. Passage in this paper represents a sentence or a paragraph. The QG task is defined to generate the question $\overline{q}$,

$$\overline{q} = \arg\max_{q} P(q|p,a) \tag{1}$$

where $P(q|p,a)$ is the conditional likelihood of the predicted question q, given a passage p and an answer a. Moreover, p has m words, i.e., $p = \{x_t\}_{t=1}^{m}$.

The mainstream strategy for automatic QG is to leverage the seq-to-seq text generation networks[20]. However, the QG models using this strategy cannot generate comprehension questions that meet teaching needs.

To address the challenges, we first take the passage-level neural question generation model[5] as the baseline model and then propose a new QG model by introducing the Gated Recurrent Units (GRU) network and GAT mechanism to our QG model (see Fig. 1). First, the input passage representation is fed into an answer-oriented GAT to obtain answer-focus context embedding, i.e., GAT embedding. Second, the concatenation of word embedding, answer tagging, and GAT embedding serve as input to the bidirectional GRU encoder. Further, a gated self-attention mechanism is then employed to the passage's hidden states. On the basis of the above steps, we unify the hidden states of passage and answer to obtain answer-aware context embeddings. Finally, by employing an attention mechanism and maxout pointer-based decoding method, the decoder can generate questions word-by-word. The variables in Fig. 1 are specified in Table 1.

3.2 Passage-answer encoding with GAT

In the encoder, we use a two-layer bidirectional GRU, the hidden state h_t^p at time step t is the concatenation of the forward hidden state $\overrightarrow{h_t^p}$ and backward hidden state $\overleftarrow{h_t^p}$, i.e., $h_t^p = [\overrightarrow{h_t^p}, \overleftarrow{h_t^p}]$, and all hidden states can be represented as $H^p = \{h_t^p\}_{t=1}^{m}$. GRU is a variant of LSTM with fewer parameters and better performance. GRU takes a passage and many answers as input, then outputs the passage-answer embedding representations. The hidden state of the t-th passage-word is h_t^p,

$$h_t^p = GRU\left(h_{t-1}^p, x_t^p\right) \tag{2}$$

where h_{t-1}^p represents the hidden state of the $(t-1)$-th word of the input passage; x_t^p is a passage word at time t.

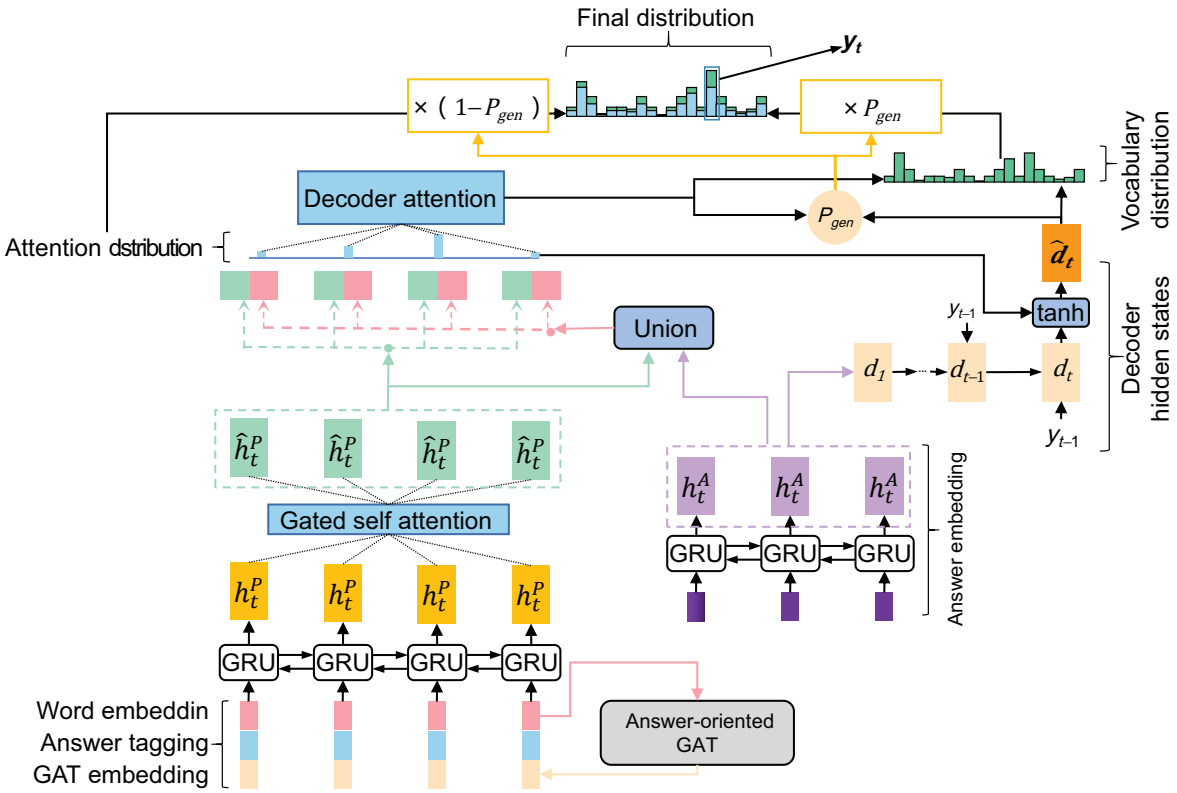

Fig. 1 End-to-end framework for our model.

Table 1 Description of variables in Fig. 1.

Variable	Description
h_t^P	Hidden state of the t-th passage-word
$\widehat{h}_t^P$	Updated hidden state of the t-th passage-word
h_t^A	Hidden states of the t-th answer-word
d_{t-1}	Hidden state of the decoder at time step $t-1$
d_t	Hidden state of the decoder at time step t
$\widehat{d}_t$	Updated hidden state of the decoder at time step t
y_{t-1}	Decoder's prediction word at time step $t-1$
y_t	Decoder's prediction word at time step t
P_{gen}	Used to decide whether the predicted word is copied from the input sequence or generated from the vocabulary

3.2.1 Answer tagging

To construct our RACE4QG dataset, the original RACE dataset is reconstructed to meet the QG task. Given that RACE is a QA dataset, its questions and answers are selected from the actual English exams, and answer words are scattered in the context. It is different from the general QA datasets (such as SQuAD), where the answer is successive spans. In this respect, traditional answer tagging methods[5, 6] are dysfunctional for our task. Although Jia et al.[7] proposed a keyword-tagging strategy, it is a hard matching strategy in which only the exact matching passage words can be tagged. Any passage word that is highly similar to any answer word, intuitively, is also effective in guiding QG. To accomplish this, we employ a new answer tagging strategy to tag answer words in a passage. Specifically, given an answer, we first tokenize the answer and remove stop words to obtain a set of words called X. Then, we tag each passage word (word occurring in the passage) as "A" if its similarity to a word in X is greater than 0.5, and other passage words are tagged as "O".

3.2.2 Answer-oriented GAT

Our QG task aims to generate real comprehension questions that require deep reasoning within and between sentences; however, traditional QG models based on RNN cannot meet this requirement.

To tackle the challenge, we can employ GAT, a type of graph neural network[21], to capture the complex

relations either intra-sentence or inter-sentence[22]. We propose an AO-GAT to encode the input passage. Following the work of Zhang et al.[23], the AO-GAT is constructed through the following steps:

(1) Answer-oriented graph adjacency matrix. To capture the internal dependency information within a passage, we build a dependency graph for a specified passage; then, the graph is represented as an adjacency matrix. The workflow is as follows:

First, the StanfordCoreNLP library[24] is utilized to generate a parsing dependency for each sentence per passage. Then each sentence corresponds to a sentence-level parsing dependency tree, i.e., S-tree.

Second, if two adjacent S-trees have answer words (words of answer text), then connect the two S-trees to build a passage-level dependency parsing graph. In the graph, each node represents a passage word.

Third, we construct a corresponding adjacent matrix G, where $G_{i,j}$ is 1 if node i and node j are connected, and 0 otherwise.

(2) GAT mechanism applied to the graph adjacency matrix for getting richer node features. Based on the adjacency matrix G, the implementation of GAT follows the work of Velickovic et al.[22] In the encoding stage, the intra-sentence and inter-sentence dependency information is stored in the graph adjacency matrix G. In our QG model, we adopt the GAT[22] to extract this information from G.

Algorithm 1 gives the detaited procedure of using GAT to capture the structure information in the passage. The major work is to update the embedding of the node by using the attention-weighted sum of the embeddings of its neighboring words. The embeddings of node i, j, and k are denoted as e_i, e_j, and e_k, respectively. The embeddings of node i's neighbors are denoted as N_i, and G represents the graph adjacency matrix.

Algorithm 1 Capture the structure information of a passage

1: **for** each node i of G **do**
2: **for** each neighboring node j of node i **do**
3: Attention of node i over its neighbor j is $a_{i,j}$,

$$a_{i,j} = \frac{\exp\left(e_i, e_j\right)}{\sum_{k \in N_i} \exp\left(e_i, e_k\right)},$$

4: **end for**
5: Embedding of node i ,namely e_i, is again represented by its neighbors,

$$e_i' = \text{sigmoid}\left(\sum_{j \in N_i} a_{i,j} e_i\right),$$

6: **end for**
7: Final output of GAT is e',

$$e' = \left\{e_1', e_2', \ldots, e_m'\right\},$$

8: e' becomes one of the inputs to the framework's encoder.

The GAT procedure comprises three steps: input, calculation, and output. The GAT takes both the passage's word embedding and the adjacency matrix G as input in the input step. The dependency information in the input passage is captured in the calculation step by calculating each node's attention against its neighbors. The GAT feeds its output (a higher-dimensional word embedding, i.e., the GAT embedding) into the framework's encoder in the output step. The specific steps are as follows:

Step 1: Input G and e to the GAT. G is the adjacent matrix, e is a set of features (i.e., a set of passage words' embeddings), where $e = \{e_1, e_2, \ldots, e_m\}$, $e_i \in \mathbf{R}^F$, and F represents the number of features of each node.

Step 2: Calculate the attention of node i against each of its neighbors,

$$a_{ij} = \frac{\exp\left(\text{PReLU}\left(\left(W^a\right)^{\mathrm{T}}\left[e_i, e_j\right]\right)\right)}{\sum\limits_{k \in N_i} \exp\left(\text{PReLU}\left(\left(W^a\right)^{\mathrm{T}}\left[e_i, e_k\right]\right)\right)} \tag{3}$$

where W^a is a trainable weight matrix; N_i represents all neighboring nodes of node i. The original article[22] used LeakyReLU as the activation function. In our experiment, we compared various activation functions, such as ELU, ReLU, ReLU6, and PReLU, and found that PReLU performs better in our QG task.

Step 3: Feed the GAT output to the encoder. Given a set of node features $e = \{e_1, e_2, \ldots, e_m\}$, $e_i \in \mathbf{R}^F$, the GAT will take the set e as input to obtain a corresponding output (of potentially different cardinality F') $e' = \left\{e_1', e_2', \ldots, e_m'\right\}$, $e_i' \in \mathbf{R}^{F'}$. Next, we will expand the calculation process of e',

$$e_i' = \text{sigmoid}\left(\sum_{j \in N_i} a_{ij} e_i\right) \tag{4}$$

To further improve the performance of self-attention, we utilize the multi-head attention mechanism[25]. Each attention head is responsible for capturing the particular features in the passage, then features captured by all attention heads are concatenated as the output feature representation. In this case, Eq. (4) becomes

$$e_i' = \text{sigmoid}\left(\sum_{k=1}^{K} \sum_{j \in N_i} a_{ij}^k W^k e_j\right) \tag{5}$$

where W^k is a trainable weight matrix, K is the number of attention heads. Our experimental results show that $k = 8$ works best. As the final output of GAT, $e' =$

$\{e'_1, e'_2, \ldots, e'_m\}$ will be concatenated with the word embedding and answer tagging as the encoder's input.

3.2.3 Gated self-attention

After the above steps, we obtain the raw passage-answer representation. To further aggregate information from the passage and incorporate intra-passage dependency to improve the passage-answer embedding at each time step, a gated self-attention mechanism is first applied to obtain self-matching representation.

$$a_t^E = \text{softmax}((H^p)^{\mathrm{T}} W^s h_t^p) \tag{6}$$

$$S_t = H^p \times a_t^E \tag{7}$$

where a_t^E represents the attention coefficient of t-th word in the input passage; W^s is a trainable weight matrix, S_t is the weighted sum of all passage words' embeddings based on their attention to the current word at time step t.

Then, combine the raw representation h_t^p and the self-matching representation S_t to obtain a new passage-answer representation $\widehat{h}_t^p$,

$$f_t = \tanh(W^f [h_t^p, S_t]) \tag{8}$$

$$g_t = \text{sigmoid}(W^g [h_t^p, S_t]) \tag{9}$$

$$\widehat{h}_t^p = g_t \times f_t + (1 - g_t) \times h_t^p \tag{10}$$

where f_t is the new self matching enhanced representation; W^f and W^g are trainable weight matrixes; g_t is a gate vector for selecting information between h_t^p and S_t to obtain the final passage-answer representation $\widehat{h}_t^p$.

3.2.4 Passage-answer union

Intuitively, the information between passage words and answer words can serve to the generation of good questions. Thus, we unify the passage raw hidden states H^P and the answer hidden states H^A as

$$H^u = \text{union}(W^u[\widehat{H}^P; H^A; \widehat{H}^P \times H^A]) \tag{11}$$

where W^u is a trainable weight matrix; $\widehat{H}^P = \{\hat{h}_t^p\}_{t=1}^m$, $H^A = \{h_t^A\}_{t=1}^n$, n and m are the number of answer words and passage words, respectively.

3.3 Decoding with attention mechanism and pointer network

The decoder (a single-layer unidirectional GRU) is trained to predict the next word y_t. On each time step t, an attention mechanism is applied to the encoder's final hidden states to highlight the more important words from the passage, then we can obtain a dynamic representation of the original text, called the context vector C_t. Then, the concatenation of C_t, previous words $(y_1, y_2, \ldots, y_{t-1})$ emitted by the decoder, and the current decoder state d_t will be fed into the decoder to generate the next word y_t by using the pointer network.

3.3.1 Attention mechanism

Attention mechanism[26, 27] has been a default setting for improving the performance of seq-to-seq models. In our decoder, the Luong attention mechanism[27] is used to obtain the raw attention a_t^D,

$$a_t^D = \text{softmax}(\widehat{H}^P W^a d_t) \tag{12}$$

$$C_t = \widehat{H}^p a_t^D \tag{13}$$

$$\widehat{d}_t = \tanh(W [d_t, C_t]) \tag{14}$$

$$d_{t+1} = GRU([y_t, \widehat{d}_t]) \tag{15}$$

an attention layer Eq. (14) is applied over the concatenation of the decoder state d_t and the attentive context vector C_t to obtain a new decoder state $\hat{d}_t$. Then, $\hat{d}_t$ can be used to generate its next state d_{t+1}.

3.3.2 Pointer network

In the QG task of this paper, the output vocabulary of the decoder must change dynamically according to the length of the input sequence to generate better questions. However, the traditional seq-to-seq model cannot tackle the challenge. To this end, we follow the work of See et al.[28] to address this issue using pointer networks. On each step t of the decoder, P_{gen} is calculated from the context vector C_t, the new decoder state $\widehat{d}_t$ and the decoder's previous output y_{t-1},

$$p_{gen} = \sigma(W_h^{\mathrm{T}} \times C_t + W_d^{\mathrm{T}} \times \widehat{d}_t + W_y^{\mathrm{T}} \times y_{t-1}) \tag{16}$$

$$p_{vocab} = \text{softmax}(W[h_t^D, C_t]) \tag{17}$$

$$p_{copy} = \max_{x_j = y_t} a_j^t \tag{18}$$

where p_{gen} is a trainable parameter, which is used to decide whether the predicted word is copied from the input sequence or generated from the vocabulary. p_{vocab} represents the generated probability from the vocabulary, and p_{copy} represents the copy probability from the input passage. On the basis of p_{gen}, p_{copy}, and p_{vocab}, we can calculate the probability distribution of the decoder's output word,

$$p(y_t \mid y_1, \ldots, y_{t-1}) = p_{vocab} \times p_{gen} + p_{copy} \times (1 - p_{gen}) \tag{19}$$

4 Experiment

4.1 Dataset

We evaluate our model on a QG dataset RACE4QG which is adapted from the RACE dataset[11]. The RACE dataset is a large-scale exam-type dataset for question answering, and the dataset was released by

Carnegie Mellon University in 2017. RACE includes 27 933 passages with 97 687 questions from the English examinations for Grades 7–12 students in China. In RACE, each sample is a quadruple of (passage, answer, question, distractor). Here a distractor is a wrong answer.

To accommodate the QG task, we must adjust RACE following the strategy of the EQG-RACE dataset[7]. First, distractors must be removed. Distractors are wrong answers, which are designed to puzzle students and may bring noise when generating good questions. Second, only questions related to the QG task can be reserved. After some detailed investigation of the RACE dataset, we observe that the questions in the RACE dataset fall into two categories: cloze-type questions and standard questions. Cloze-type question is also called the fill-in-the-blank question, which belongs to the traditional QA task and cannot be used directly for our QG task. Therefore, to construct the RACE4QG dataset based on the RACE dataset, we first remove the cloze-type questions, then use the answer tagging strategy mentioned above to tag the answer words from passages. Finally, the RACE4QG dataset has 46 397 samples, accounting for 47.5% of all RACE's samples. As a result, each sample is a triple of (passage, answer, question). Our task is to generate questions on the basis of the given passages and answers. Compared to the previous EQG-RACE dataset, our RACE4QG has more than twice as many questions as EQG-RACE does. Further, our answer tagging method innovatively introduces a similar scheme (see Section 3.2 for details).

To train the model, the RACE4QG dataset must be further divided into three splits. In the original RACE dataset, there are 87 866, 4887, and 4934 samples in the training set, validation set, and test set, respectively. In contrast, our RACE4QG has 41 791, 2312, and 2294 samples for training, validation, and testing, respectively.

4.2 Implementation details

The vocabulary of our model contains 4.5×10^4 tokens, and the pretrained GloVe.840B.300d[29] is utilized as the initialization of word embedding. Other out-of-vocabulary tokens are set to the UNK symbol.

In our model, three types of GRU are used in three places. First, the encoder uses a two-layer bidirectional GRU with a hidden unit size of 600 (300 in each direction). Second, the decoder uses a single-layer unidirectional GRU with a hidden unit size of 300. Third, the answer encoder uses a single-layer bidirectional GRU with a hidden unit size of 600 (half in each direction).

The dropout probability is set to 0.3. All trainable parameters are set to $U(-0.1, 0.1)$, except for word embedding. During training, the stochastic gradient descent optimizer was used, with a batch size of 45 and an initial learning rate of 0.01 for our model and baselines. In our model, the learning rate was fixed at 0.01 for the first eight epochs, then it was halved every other epoch, but could not be less than 0.001.

In the decoder, the beam size is set to 10. Models' checkpoints are chosen on the validation set, and we report the results on the test set.

4.3 Baseline

To evaluate the performance of our model against other baselines, we rewrite the code of five famous baselines. (1) Seq-to-seq[30]: A seq-to-seq model using attention-mechanism and copy-policy. (2) Pointer-generator[31]: Answer-focused and position-aware model for QG using the pointer-generator mechanism. (3) Transformer[32]: Transformer-based end-to-end QG. (4) ELMo-QG[33]: Apply a pointer network to copy words from the input. (5) AG-GCN[7]: A QG model with a GCN-based encoder using LSTM.

For fairness, our answer tagging policy is also applied to the baselines. Further, the encoder of each baseline uses a two-layer bidirectional LSTM, and the decoder is a one-layer unidirectional LSTM. The remaining settings are remained untouched.

5 Result and Analysis

5.1 Automatic evaluation

We will automatically evaluate the similarity between the generated questions and real questions. To carry out evaluation tasks comprehensively, we select metrics from the perspectives of precision, recall, and semanteme. To this end, we employ BLEU-(1–4)[34], ROUGE-L[35], and METEOR[36]. BLEU evaluates n-gram precision between the generated questions and the real questions. ROUGE-L is responsible for evaluating the recall rate. Note that the above two metrics belong to literal similarity. However, the evaluation with semantic similarity must be introduced, so we employ the third metric, METEOR.

In Table 2, the evaluation results are listed for our model and baselines. By using the GAT mechanism and GRU network, our model outperforms baselines across all metrics. In addition, a clear performance gap exists between the two baselines (i.e., seq-to-seq and Transformer). The main reason could be that the

Table 2 Comparative experimental results of different models.

Model	Evaluation metrics					
	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-L	METE0R
Seq-to-seq[30]	24.97	10.48	6.46	5.35	25.61	10.11
Pointer-generator[31]	30.47	14.57	8.89	6.59	31.81	13.8
Transformer[32]	30.44	15.44	9.52	6.85	34.22	15.03
ELMo-QG[33]	34.46	17.79	12.00	8.83	34.05	14.89
AG-GCN[7]	35.59	21.09	15.21	12.12	34.71	15.14
Our model	**36.61**	**21.32**	**15.26**	**12.56**	**36.03**	**16.48**

Transformer uses the same hierarchical architectures as our model. In addition, our model performs better than Transformer, which indicates that the GAT mechanism plays a vital role in capturing information from intra-sentence and inter-sentence. Finally, our model with GAT+GRU is better than AG-GCN[7] with GCN+LSTM in key metrics. We also compared the Jaccard distance of 10 questions generated by the beam search algorithm, and we found that the quality of these questions decreased sequentially. The reason may be that the latter questions have a lower likelihood.

In Table 3, we show the superiority of our dataset RACE4QG over the previous dataset EQG-RACE; we train our model on RACE4QG and EQG-RACE, separately. Experimental results show that our dataset RACE4QG can perform better than EQG-RACE.

In Table 4, to assess the effectiveness of both the GAT mechanism and GRU network, we conduct two types of ablation experiments. First, our model has various degrees of performance degradation after removing GAT and GRU. Second, our model uses the style of GAT+GRU, compared to the GCN+LSTM style of the previous AG-GCN. Table 4 shows that our GAT mechanism is superior to GCN, which indicates that GAT can better capture the important semantic and syntactic information within and between sentences. Similarly, if the encoder's GRU is replaced with a traditional LSTM, the performance of our model also drops because the GRU fits our dataset better.

5.2 Human evaluation

To assess the performance between the generated questions and the real questions, the human evaluation is utilized to compare the baseline (i.e., seq-to-seq model) with our model. We invited five evaluators to score [0, 1, 2] the generated questions on the basis of the following three metrics:

- **Fluency:** The grammatical and structural fluency of the generated questions.
- **Answerability:** The extent to which the generated questions can be answered.
- **Comprehension:** The level of reading comprehension can be tested by the generated questions. The score of comprehension falls between 0 and 2. The higher the score, the higher the reading level. A question with a higher comprehension score means that the question assesses a student's higher-level reading comprehension.

After obtaining the scores of the five evaluators, we averaged the scores and placed them in Table 5. The results in Table 5 show that our model has a clear advantage over the baseline. In addition, we observe that the comprehension metrics are all lower, which indicates that the generated questions are not enough to meet the teaching needs.

Table 3 Superiority of our RACE4QG dataset.

Dataset	Evaluation metrics					
	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-L	METE0R
Previous EQG-RACE[7]	36.1	21.02	15.04	12.38	35.53	16.25
Our RACE4QG	**36.61**	**21.32**	**15.26**	**12.56**	**36.03**	**16.48**

Table 4 Results of the ablation experiments for our model.

Method	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-L	METE0R
All=GRU+GAT	36.61	21.32	15.26	12.56	36.03	16.48
GRU+GCN	35.66	18.68	11.61	8.03	35.34	15.90
LSTM+GAT	36.11	21.24	15.24	12.36	35.43	15.97
GRU	34.25	17.31	10.34	6.92	34.52	14.75
LSTM+GCN	35.60	21.16	15.21	12.16	34.84	15.45

Table 5 Results of the human evaluation for the generated questions between our model and the baseline (seq-to-seq).

Model	Metrics		
	Fluency	Answerability	Comprehension
Baseline	1.62	1.47	0.8
Our model	**1.76**	**1.76**	**1.1**

5.3 Case study

To demonstrate our QG tasks in practice, we generated questions based on a passage from our dataset. The generated and actual questions are displayed in Table 6. In Table 6, given a passage and three real questions, our model generated a fourth and fifth question in addition to the three questions that correspond to the real question. The first question generated by our model closely resembles the first actual question but contains ambiguities. The quality of the second and third questions is high. Additionally, our model generated two additional questions, numbered 4 and 5.

Passage: What is your idea of a good time? What about dancing in a rainy field with 150 000 other people while a famous rock band plays on a stage so far away that the performers look like ants? It may sound strange, but that is what many hundreds of thousands of young people in the UK do every summer. Why? Because summer is the time for outdoor music festivals. Held on a farm, the Glastonbury Festival is the most well-known and popular festival in the UK. It began in 1970 and the first festival was attended by one thousand five hundred people, each paying an admission price of PS1—the ticket included free milk from the farm. Since then, the Glastonbury Festival has gone from strength to strength—in 2004, one hundred and fifty thousand fans attended, paying PS112 each for a ticket to the three-day event. Tickets for the event sold out within three hours. Performers included superstars such as Paul McCartney and James Brown, as well as new talents like Franz Ferdinand and Joss Stone. Although man, Glastonbury is a charity event, donating millions of pounds to local and international charities. Glastonbury is not unique in using live music to raise money to fight global poverty. In July of this year, eight live concerts were held simultaneously in London, Paris, Rome, and Berlin. Superstars such as Madonna, Sir Elton John, and Stevie Wonder performed to highlight international poverty and debt.

6 Conclusion

In the teaching of reading comprehension, the gains of teaching are limited by a long-standing pain point, where teachers are unable to automatically and timely generate questions based on arbitrary passages (i.e., reading comprehension materials). For this reason, we propose an end-to-end QG model that is trained on a reconstructing dataset called RACE4QG. To enrich the passage-answer embedding representations, we introduce an AO-GAT mechanism combined with a gated self-attention approach and a union strategy in the model's encoder. In addition, the performance of the model's decoder is enhanced by introducing an attention mechanism and pointer network. Experimental results show that our model outperforms baselines in both automatic evaluation and human evaluation.

However, a gap exists between experimental results and the expected results of humans. There may be two reasons. First, our dataset is inadequate in size. To construct RACE4QG, we remove 52.5% of Cloze-style questions (i.e., fill-in-the-blank questions) from the original RACE dataset. Therefore, our future efforts will concentrate on converting the cloze-style questions into standard questions, thereby doubling the size of our RACE4QG. Besides, utilizing recent advances in text representation—deep language representation—can further enhance the performance of our model[19].

Acknowledgment

This work was supported by the National Natural Science Foundation of China (No. 62166050), Yunnan Fundamental Research Projects (No. 202201AS070021), Yunnan Innovation Team of Education Informatization for Nationalities, Scientific Technology Innovation Team of Educational Big Data Application Technology in University of Yunnan Province, and Yunnan Normal University Graduate Research and innovation fund in 2020 (No. ysdyjs2020006).

Table 6 Case study of our generated questions.

Real question	Generated question
Question 1: What did the performers look like?	Question 1: What is the name of a rock band that plays on a stage?
Question 2: How many people attended the second festival?	Question 2: How many fans attended the Glastonbury Festival in 2004?
Question 3: Where were the eight live concerts held in July 2014?	Question 3: Where were the eight live concerts held in July of this year?
–	Question 4: What is the time for outdoor music festivals?
–	Question 5: How many people attended the first Glastonbury Festival?

References

[1] B. Ghanem, L. L. Coleman, J. R. Dexter, S. von der Ohe, and A. Fyshe, Question generation for reading comprehension assessment by modeling how and what to ask, in *Proc. Findings of the Association for Computational Linguistics: ACL 2022*, Dublin, Ireland, 2022, pp. 2131–2146.

[2] P. D. Pearson, The roots of reading comprehension instruction, in *Handbook of Research on Reading Comprehension*, S. E. Israel and G. G. Duffy, eds. New York, NY, USA: Routledge, 2009, pp. 3–31.

[3] X. Du, J. Shao, and C. Cardie, Learning to ask: Neural question generation for reading comprehension, in *Proc. 55^{th} Annu. Meeting of the Association for Computational Linguistics* (*Volume 1: Long Papers*), Vancouver, Canada, 2017, pp. 1342–1352.

[4] G. Kurdi, J. Leo, B. Parsia, U. Sattler, and S. Al-Emari, A systematic review of automatic question generation for educational purposes, *International Journal of Artificial Intelligence in Education*, vol. 30, no. 2, pp. 121–204, 2020.

[5] Y. Zhao, X. Ni, Y. Ding, and Q. Ke, Paragraph-level neural question generation with maxout pointer and gated self-attention networks, in *Proc. 2018 Conf. on Empirical Methods in Natural Language Processing*, Brussels, Belgium, 2018, pp. 3901–3910.

[6] Q. Zhou, N. Yang, F. Wei, C. Tan, H. Bao, and M. Zhou, Neural question generation from text: A preliminary study, in *Proc. 6^{th} CCF Conf. on Natural Language Processing and Chinese Computing*, Dalian, China, 2018, pp. 662–671.

[7] X. Jia, W. Zhou, X. Sun, and Y. Wu, EQG-RACE: Examination-type question generation, in *Proc. 35^{th} AAAI Conf. on Artificial Intelligence*, Virtual Event, 2021, pp. 13143–13151.

[8] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, SQuAD: 100 000+ questions for machine comprehension of text, in *Proc. 2016 Conf. on Empirical Methods in Natural Language Processing*, Texas, TX, USA, 2016, pp. 2383–2392.

[9] T. Kočiský, J. Schwarz, P. Blunsom, C. Dyer, K. M. Hermann, G. Melis, and E. Grefenstette, The NarrativeQA reading comprehension challenge, *Trans. Assoc. Comput. Linguist.*, vol. 6, pp. 317–328, 2018.

[10] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. Cohen, R. Salakhutdinov, and C. D. Manning, HotpotQA: A dataset for diverse, explainable multi-hop question answering, in *Proc. 2018 Conf. on Empirical Methods in Natural Language Processing*, Brussels, Belgium, 2018, pp. 2369–2380.

[11] G. Lai, Q. Xie, H. Liu, Y. Yang, and E. Hovy, RACE: Large-scale reading comprehension dataset from examinations, in *Proc. 2017 Conf. on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark, 2017, pp. 785–794.

[12] S. Hochreiter and J. Schmidhuber, Long short-term memory, *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[13] K. Dhole and C. D. Manning, Syn-QG: Syntactic and shallow semantic rules for question generation, in *Proc. 58^{th} Annu. Meeting of the Association for Computational Linguistics*, Virtual Event, 2020, pp. 752–765.

[14] V. Rus, B. Wyse, P. Piwek, M. Lintean, S. Stoyanchev, and C. Moldovan, Question generation shared task and evaluation challenge-status report, in *Proc. 13^{th} European Workshop on Natural Language Generation*, Nancy, France, 2011, pp. 318–320.

[15] M. Heilman and N. A. Smith, Good question! Statistical ranking for question generation, in *Proc. Human Language Technologies: The 2010 Annu. Conf. of the North American Chapter of the Association for Computational Linguistics*, Los Angeles, CA, USA, 2010, pp. 609–617.

[16] I. Labutov, S. Basu, and L. Vanderwende, Deep questions without deep understanding, in *Proc. 53^{rd} Annu. Meeting of the Association for Computational Linguistics and the 7^{th} Int. Joint Conf. on Natural Language Processing* (*Volume 1: Long Papers*), Beijing, China, 2015, pp. 889–898.

[17] X. Yao, G. Bouma, and Y. Zhang, Semantics-based question generation and implementation, *Dialogue & Discourse*, vol. 3, no. 2, pp. 11–42, 2012.

[18] L. Song, Z. Wang, and W. Hamza, A unified query-based generative model for question generation and question answering, arXiv preprint arXiv: 1709.01058, 2017.

[19] W. Yuan, T. He, and X. Dai, Improving neural question generation using deep linguistic representation, in *Proc. Web Conf. 2021*, Ljubljana, Slovenia, 2021, pp. 3489–3500.

[20] T. Hosking and S. Riedel, Evaluating rewards for question generation models, in *Proc. 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1* (*Long and Short Papers*), Minneapolis, MN, USA, 2019, pp. 2278–2283.

[21] X. Zhou, W. Liang, W. Li, K. Yan, S. Shimizu, and K. I. K. Wang, Hierarchical adversarial attacks against graph-neural-network-based IoT network intrusion detection system, *IEEE Internet Things J.*, vol. 9, no. 12, pp. 9310–9319, 2022.

[22] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, Graph attention networks, presented at the 6th Int. Conf. on Learning Representations, Vancouver, Canada, 2018.

[23] Y. Zhang, P. Qi, and C. D. Manning, Graph convolution over pruned dependency trees improves relation extraction, in *Proc. 2018 Conf. on Empirical Methods in Natural Language Processing*, Brussels, Belgium, 2018, pp. 2205–2215.

[24] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, The Stanford CoreNLP natural language processing toolkit, in *Proc. of 52^{nd} Annu. Meeting of the Association for Computational Linguistics: System Demonstrations*, Baltimore, MD, USA, 2014, pp. 55–60.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, Attention is all you need, in *Proc. 31^{st} Int. Conf. on Neural Information Processing Systems*, Long Beach, CA, USA, 2017, pp. 6000–6010.

[26] D. Bahdanau, K. Cho, and Y. Bengio, Neural machine translation by jointly learning to align and translate, presented at the 3rd Int. Conf. on Learning Representations, San Diego, CA, USA, 2015.

[27] M. T. Luong, H. Pham, and C. D. Manning, Effective approaches to attention-based neural machine translation, in *Proc. 2015 Conf. on Empirical Methods in Natural Language Processing*, Lisbon, Portugal, 2015, pp. 1412–1421.

[28] A. See, P. J. Liu, and C. D. Manning, Get to the point: Summarization with pointer-generator networks, in *Proc. 55th Annu. Meeting of the Association for Computational Linguistics* (*Volume 1*: *Long Papers*), Vancouver, Canada, 2017, pp. 1073–1083.

[29] J. Pennington, R. Socher, and C. D. Manning, GloVe: Global vectors for word representation, in *Proc. 2014 Conf. on Empirical Methods in Natural Language Processing* (*EMNLP*), Doha, Qatar, 2014, pp. 1532–1543.

[30] B. Liu, Neural question generation based on Seq2Seq, in *Proc. 5th Int. Conf. on Mathematics and Artificial Intelligence*, Chengdu, China, 2020, pp. 119–123.

[31] X. Sun, J. Liu, Y. Lyu, W. He, Y. Ma, and S. Wang, Answer-focused and position-aware neural question generation, in *Proc. 2018 Conf. on Empirical Methods in Natural Language Processing*, Brussels, Belgium, 2018, pp. 3930–3939.

[32] L. E. Lopez, D. K. Cruz, J. C. B. Cruz, and C. Cheng, Simplifying paragraph-level question generation via transformer language models, in *Proc. 18th Pacific Rim Int. Conf. on Artificial Intelligence*, Hanoi, Vietnam, 2021, pp. 323–334.

[33] S. Zhang and M. Bansal, Addressing semantic drift in question generation for semi-supervised question answering, in *Proc. 2019 Conf. on Empirical Methods in Natural Language Processing and the 9th Int. Joint Conf. on Natural Language Processing* (*EMNLPIJCNLP*), Hong Kong, China, 2019, pp. 2495–2509.

[34] K. Papineni, S. Roukos, T. Ward, and W. J. Zhu, Bleu: A method for automatic evaluation of machine translation, in *Proc. 40th Annu. Meeting of the Association for Computational Linguistics*, Philadelphia, PA, USA, 2002, pp. 311–318.

[35] C. Y. Lin, ROUGE: A package for automatic evaluation of summaries, presented at the Workshop on Text Summarization Branches Out, Post-Conf. Workshop of ACL 2004, Barcelona, Spain, 2004.

[36] S. Banerjee and A. Lavie, METEOR: An automatic metric for MT evaluation with improved correlation with human judgments, in *Proc. Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, Ann Arbor, MI, USA, 2005, pp. 65–72.

Jian Xu received the MEng degree in computer software and theory from Yunnan Normal University, Kunming, China in 2008. Now he is a PhD candidate in educational technology at Yunnan Normal University, Kunming, China. He is a professor at Qujing Normal University, Qujing, China. His research interests include natural language processing and smart education.

Yu Sun received the PhD degree in computer software and theory from Chinese Academy of Sciences, Beijing, China in 2006. She is a professor at Yunnan Normal University, Kunming, China. Her main research inrterest is intelligent teaching environment.

Jianhou Gan received the PhD degree from Kunming University of Science and Technology, Kunming, China in 2016. He is a professor at Yunnan Normal University, Kunming, China. His main research interest is smart education.

Mingtao Zhou received the BEng degree from Anyang Institute of Technology, Anyang, China in 2020. Now he is a master student in educational technology at Yunnan Normal University, Kunming, China. His research interests include deep learning and natural language processing.

Di Wu received the MEng degree from Yunnan Normal University, Kunming, China in 2021. Now he is a PhD candidate in educational technology at Yunnan Normal University, Kunming, China. His research interests include machine learning and natural language processing.