Iterative GNN-based Decoder for Question Generation

Zichu Fei, Qi Zhang, Yaqian Zhou
Shanghai Key Laboratory of Intelligent Information Processing
School of Computer Science, Fudan University, Shanghai, China
{zcfei19, qz, yqzhou}@fudan.edu.cn

Abstract

Natural question generation (QG) aims to generate questions from a passage, and generated questions are answered from the passage. Most models with state-of-the-art performance model the previously generated text at each decoding step. However, (1) they ignore the rich structure information that is hidden in the previously generated text. (2) they ignore the impact of copied words on the passage. We perceive that information in previously generated words serves as auxiliary information in subsequent generation. To address these problems, we design the Iterative Graph Network-based Decoder (IGND) to model the previous generation using a Graph Neural Network at each decoding step. Moreover, our graph model captures dependency relations in the passage that boost the generation. Experimental results demonstrate that our model outperforms the state-of-the-art models with sentence-level QG tasks on SQuAD and MARCO datasets.

1 Introduction

Automatic Question Generation (QG) is the task of generating question-answer pairs from a declarative sentence, QG has many useful applications: (1) it improves the question answering task (Chen et al., 2017) by providing more training data (Tang et al., 2017; Yuan et al., 2017); (2) it generates practice exercises and assessments for educational purposes (Heilman and Smith, 2010); and (3) it helps dialog systems to kick-start and continue a conversation with human users (Mostafazadeh et al., 2016). In this study, we focus on sentence-level QG tasks.

Conventional QG methods (Mostow and Chen, 2009; Heilman and Smith, 2010; Dhole and Manning, 2020) rely on heuristic rules or hand-crafted templates as it suffers a significant lack of question, sticking to a few simple and reliable syntactic transformation patterns. Recently, neural-based approaches to QG have achieved remarkable success, by applying large-scale reading comprehension datasets and employing the encoder-decoder framework. Most of the existing works are based on the sequence-to-sequence (Seq2Seq) network, incorporating the attention mechanism and copy mode, applied by (Zhou et al., 2018). Intuitively, connecting an answer to a passage lies at the heart of this task. (Song et al., 2018) leveraged multi-perspective matching methods and (Sun et al., 2018) proposed a position-aware model to put more emphasis on answer-surrounded context words. (Zhao et al., 2018) aggregated the paragraph-level context to provide sufficient information for question generation. (Chen et al., 2020; Liu et al., 2019) employed the Graph2Seq architecture to capture the information in a passage.

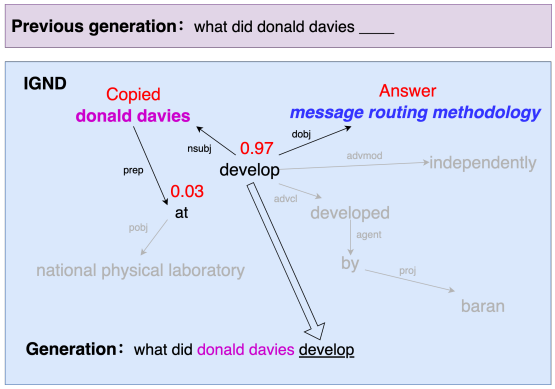

Figure 1: An example (lower-cased) of using the structure information that is hidden in previously generated words information and the impact of copied word, where the answer words are in blue and the copied words are in purple. The model can copy the right word **develop** with a high certainty (0.97 score).

Most models with state-of-the-art performance model the previously generated text at each decoding step. However, they ignore (1) the rich structure information hidden in generated words (2) and the impact of copied words on the passage. We perceive that this information offers auxiliary information in the future generation. In Figure 1, the copied word *donald davies* helps model to copy

the *develop* with high certainty (0.97 score). The copied word *donald davies* is a subject in the passage and the answer *message routing methodology* is the object in the passage. After capturing the structure information from the generated words, the model pays more attention to the words related to generated words and copy the predicate in the passage. However, the decoders in most QG model process the generated text as the sequence of words, which ignore the text structure. Therefore, it is hard to capture the structure information in previously generated words for most QG models. In addition, the information about which word has been copied changes at each step and is updated iteratively. However, most QG models are unable to achieve that.

To address these issues, in this paper, we design an Iterative Graph Network-based Decoder (IGND) to model the structure information in the previous generation at each decode step using a Graph Neural Network. We observed that the words copied from a passage played a decisive role in the semantics of the whole question. We modeled the copied word information to capture structure information and use their impact on the passage. We introduce the role tag to the passage graph, where all words have the role tag *no-copy*, except for answer words, which have the tag *answer*. The IGND updates the role tag at each decoding step. For example, the role tag changes to *copied* when the word in this node is copied to the question at this decoding step. Then, the information is aggregated by a novel Bi-directional Gated Graph Neural Network (bi-GGNN). Moreover, we propose a relational-graph encoder, which employs a similar bi-GGNN to capture the dependency relations of a passage that boost the generation.

We performed experiments on two reading comprehension datasets, SQuAD and MARCO, and obtained promising results. Our model achieves new state-of-the-art results in sentence-level QG tasks on both datasets, with BLEU-4 20.33 on SQuAD and 23.87 on MARCO.

Our main contributions are as follows:

- We design an Iterative Graph Network-based Decoder (IGND) to capture the structure information in generation and model the copied words at each decoding step.
- We propose a relational-graph encoder to encode the dependency relations in the passages and establish the connections between an answer and a passage.
- The proposed model that focuses on sentence-level QG tasks achieves new state-of-the-art scores, and outperforms existing methods on the standard SQuAD and MARCO benchmarks for QG.

2 Model Description

In this section, we define the question generation task and present the Graph-to-Sequence (Graph2Seq) model with our IGND. We design and discuss the details of each component as shown in Figure 2.

2.1 Problem Formulation

The question generation generates natural language questions based on given sentences (Zhou et al., 2018). The generated questions must be answered from the input data.

We assume that a text passage is a sequence of word tokens $X^p = \{x_1^p, x_2^p, ..., x_N^p\}$, and a target answer is a sequence of word tokens $X^a = \{x_1^a, x_2^a, ..., x_L^a\}$. The natural question generation task generates the best natural language question consisting of a sequence of word tokens $\hat{Y} = \{y_1, y_2, ..., y_T\}$ and maximizing the conditional likelihood $\arg\max_Y P(Y|X^p, X^a)$. Here N, L, and T are the lengths of the passage, the answer, and the question, respectively. We focus on the problem set based on a set of passages, answers, and questions triples. We learn the connection between them. Existing QG approaches (Zhou et al., 2018; Sun et al., 2018; Song et al., 2018; Zhao et al., 2018; Chen et al., 2020) have the same assumption.

2.2 Graph2Seq Model with Iterative Graph Network-based Decoder

Compared to RNNs, GNNs can efficiently use the rich hidden text structure information such as syntactic information. In addition, they can model the global relations among the sequence words to improve the representations. We construct a directed and weighted text graph G based on dependency tree. In a passage graph, each passage word is treated as a node and the dependency relation between two words is treated as an edge. Furthermore, our Graph2Seq model encodes the passage graph with dependency relations and decodes the question sequence with IGND.

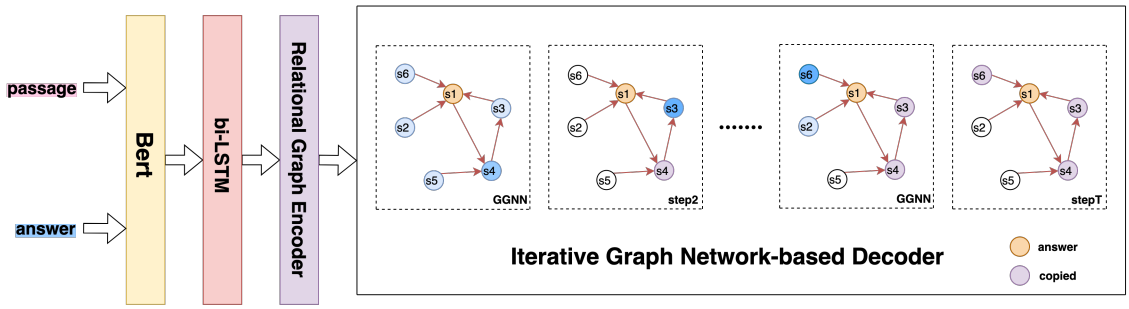

Figure 2: Overall architecture of the proposed model. In Iterative Graph Network-based Decoder, we use the shade of the color to indicate how high the node copy score is and the color of zero-score node is white. Furthermore, answer nodes and copied nodes are yellow and purple respectively.

2.2.1 Relational Encoder

Answer information is crucial to generate the high quality and answer-relevant questions. Dependency relations connect the answer and passage words. To use the dependency relations, we propose the relational embedding that aggregates global dependency relations for each words. Intuitively, relational embedding indicates words to pay more attention.

Firstly we adopt a bi-LSTM encoder to get the context hidden states H:

$$e_i = [w_i, b_i, a_i, p_i, n_i, u_i] \quad (1)$$

$$h_i = [\overrightarrow{h_i}, \overleftarrow{h_i}] \quad (2)$$

$$\overrightarrow{h_i} = LSTM(e_i, \overrightarrow{h_{i-1}}) \quad (3)$$

$$\overleftarrow{h_i} = LSTM(e_i \overleftarrow{h_{i+1}}) \quad (4)$$

where w_i, b_i, a_i, n_i, p_i, u_i represent Glove embedding of the word, BERT embedding of the word, answer position embedding, named entity embedding, part-of-speech embedding and word case embedding as proposed by (Zhou et al., 2018), $\overrightarrow{h_i}$ and $\overleftarrow{h_i}$ is forward and backward hidden states of the ith token in passage X^P. Moreover, we get the answer hidden states H^a in a similar way.

Then, we get the answer-aware weighted context hidden states H^p:

$$e_i^a = v_a^\top tanh(W_a H^a + W_h h_i) \quad (5)$$

$$\alpha_i^a = \frac{exp(e_i^a)}{\sum_{j=1}^{N} exp(e_j^a)} \quad (6)$$

$$H^p = \alpha^a H \quad (7)$$

where $H = [h_1, h_2, ..., h_N]$ is the passage hidden states and $v_a^\top$, W_a, W_h are trainable weighted matrices.

To learn the graph embedding from the text passage graph, we adopt the novel bi-GGNN that fuses the intermediate node embeddings from incoming and outgoing directions in every iteration. In bi-GGNN, passage embeddings for each nodes are initialized to the passage embeddings H^p, and the relational embeddings are initialized randomly. The graph parameters are shared at every hop of computation. And at every node in the graph, we apply a mean aggregator to aggregate neighboring node passage embedding and get the aggregation vector:

$$h^{k+1}_{\mathcal{N}_{\vdash p_i}} = \frac{h^k_{\mathcal{N}_{\vdash p_i}} + \sum_{j \in \mathcal{N}_\vdash} h^k_{p_j}}{|\mathcal{N}_\vdash|} \quad (8)$$

$$h^{k+1}_{\mathcal{N}_{\dashv p_i}} = \frac{h^k_{\mathcal{N}_{\dashv p_i}} + \sum_{j \in \mathcal{N}_\dashv} h^k_{p_j}}{|\mathcal{N}_\dashv|} \quad (9)$$

Similarly, we get the relation embedding aggregation vector:

$$h^{k+1}_{\mathcal{N}_{\vdash rela_i}} = \frac{h^k_{\mathcal{N}_{\vdash rela_i}} + \sum_{j \in \mathcal{N}_\vdash} r_{ij}}{|\mathcal{N}_\vdash|} \quad (10)$$

$$h^{k+1}_{\mathcal{N}_{\dashv rela_i}} = \frac{h^k_{\mathcal{N}_{\dashv rela_i}} + \sum_{j \in \mathcal{N}_\dashv} r_{ij}}{|\mathcal{N}_\dashv|} \quad (11)$$

where r_{ij} is the relations embedding between node i and j.

We fuse the information aggregated in two directions at each hop:

$$h^k_{\mathcal{N}_{p_i}} = Fuse(h^k_{\mathcal{N}_{\vdash p_i}}, h^k_{\mathcal{N}_{\dashv p_i}}) \quad (12)$$

$$h^k_{\mathcal{N}_{rela_i}} = Fuse(h^k_{\mathcal{N}_{\vdash rela_i}}, h^k_{\mathcal{N}_{\dashv rela_i}}) \quad (13)$$

$$Fuse(a, b) = z \odot a + (1 - z) \odot b \quad (14)$$

where z is a gated sum of two information as fusion function:

$$z = \sigma(W_z[a; b; a \odot b; a - b] + b_z) \quad (15)$$

where $\odot$ is the component-wise multiplication and σ is a sigmoid function.

We used a GRU (Cho et al.) to update the node embedding and incorpotate the aggregation information:

$$h_{p_i}^k = GRU(h_{p_i}^{k-1}, h_{\mathcal{N}_{p_i}}^k) \quad (16)$$

$$h_{rela_i}^k = GRU(h_{rela_i}^{k-1}, h_{\mathcal{N}_{rela_i}}^k) \quad (17)$$

After n hops computation, we obtain the final context embedding, relational embedding $h_{c_i}^n$, $h_{rela_i}^n$ for node i. Then, the node embedding incorporating both text information and syntactic information is calculated as:

$$h_i^n = h_{c_i}^n + h_{rela_i}^n \quad (18)$$

Furthermore, we can get a graph-level embedding $h^{\mathcal{G}}$ with a max pool:

$$h^{\mathcal{G}} = MaxPool(W^g h_i^n) \quad (19)$$

where W^g is a trainable weighted matrix.

2.2.2 Iterative Graph Network-based Decoder

We adopt the architecture similar to other QG models, which is an attention-based LSTM decoder with a copy mechanism (Sun et al., 2018). However, most existing decoders ignore the structure information hidden in previous generated words and the impact of copied words on passage, which could serve as auxiliary information. To address this problem, we design the Iterative Graph Network-based Decoder(IGND).

The decoder takes the graph-level embedding followed by two separate fully-connected layers as initial hidden states s_0 and initial context vector c_0:

$$s_0 = tanh(W_{t2}tanh(W_{t1}h^{\mathcal{G}} + b_{t1}) + b_{t2}) \quad (20)$$

$$c_0 = s_0 \quad (21)$$

We construct a graph in decoder G_d similar to the passage graph G that adds the role tag information to node embedding. We introduce the role tag to nodes in the graph: each node has a role tag that is updated at each decode step including *answer*, *copied* and *no-copy*. The nodes that represent the answer word contain the *answer* tag at whole decode process, the nodes have been copied to question obtain the *copied* tag and other nodes obtain the *no-copy* tag:

$$tag_i = \begin{cases} 0 & x_i \ has \ been \ copied \\ 1 & x_i \ is \ a \ part \ of \ answer \\ 2 & x_i \ has \ not \ been \ copied \end{cases}$$

Intuitively, role tag can guide the model to incorporate the dependency relations to generate answer-relevant questions as shown in Figure 1.

At each decode step t, the embeddings for each nodes $h_{d_i}^t$ are reinitialized:

$$h_{d_i}^t = [h_i^n; r_i^t] \quad (22)$$

where h_i^n is the node embedding of the passage graph calculated by equation (18) and r_i^t is the embedding of role tag for node i at step t. Furthermore, we adopt a bi-GGNN and a mean aggregator to aggregate the node embeddings, similar to that of Section 2.2.1. After n hops computation, we obtain the final node embeddings $h_{d_i}^t$ in decoder graph.

For each decoding step t, the LSTM reads the embedding of the previous word w_{t-1}, previous attentional context vector c_{t-1} and previous hidden state s_{t-1} to calculate its current hidden state:

$$s_t = LSTM([w_{t-1}; c_{t-1}], s_{t-1}) \quad (23)$$

At time step t, the attention weights and the context vector are calculated as:

$$e_{t,i} = v^\top tanh(W_s s_t + W_h h_{d_i}^t) \quad (24)$$

$$\alpha_{t,i} = \frac{exp(e_{t,i})}{\sum_{j=1}^N exp(e_{t,j})} \quad (25)$$

$$c_t = \sum_{i=1}^N \alpha_{t,i} h_{d_i}^t \quad (26)$$

where α is the attention weights previous generation information.

The copy mode copies words directly from the source sequence. As the attention weights measure the relevance of each input word to the partial decoding state and incorporate the generated words information, we treat α_t as the copy probability, $P_{copy} = \alpha_t$.

Then, s_i and c_i will be fed into a two-layer feed-forward network to produce the vocabulary distribution P_{vocab}.

The final probability distribution is the combination of the two modes:

$$P(w) = p_g P_{vocab} + (1 - p_g) P_{copy} \quad (27)$$

where p_g is computed from the context vector c_t, decoder hidden states s_t and the decoder input w_t:

$$p_g = \sigma(W_g(c_t + s_t + w_t)) \quad (28)$$

where W_g is a trainable weighted matrix and σ is a sigmoid function.

We train our model by the negative log likelihood for the target sequence **y**:

$$L = \frac{1}{T}\sum_{t=1}^{T} logP(\tilde{y}_t = y_t) \quad (29)$$

3 Experimental Settings

3.1 Dataset

3.1.1 SQuAD

The SQuAD (Rajpurkar et al., 2016) dataset contains 536 Wikipedia articles and more than 100K questions from the articles of crowd-workers. Answers are provided to the questions, which are spans of tokens in the articles. We use the sentence-level data shared by (Zhou et al., 2018) [1] and there are 86,635, 8,965 and 8,964 triples correspondingly.

3.1.2 MARCO

MS MARCO datasets (Nguyen et al., 2016) [2] contains 100,000 queries with corresponding answers and passages. All questions are sampled from real anonymized user queries and context passages are extracted from real web documents. We picked a subset of MS MARCO data where answers were sub-spans within the passages to construct sentence-level dataset. That contains 4,6109, 4539 and 4539 sentence-question-answer triples for training, validation and test respectively.

3.2 Baseline Methods and Metrics

For fair comparison, we report the following recent works on sentence-level QG dataset:

NQG++ (Zhou et al., 2018): a feature-enriched Seq2Seq model.

MPQG (Song et al., 2018): uses different matching strategies to explicitly model the information between answer and context.

Answer-focused Position-aware model (Sun et al., 2018): generates an accurate interrogative word and focuses on important context words.

s2sa-at-mp-gsa (Zhao et al., 2018): employs a gated attention encoder and a maxout pointer decoder to deal with long text inputs.

ASs2s (Kim et al., 2019): proposes an answer separated Seq2Seq model by replacing the answer in the input sequence with some specific words.

To the Point Context (Li et al., 2019): extracts answer-relevant relations in the sentence and encodes both sentence and relations to capture answer-focused representations.

QG-pg (Jia et al., 2020): leverages the paraphrase information to the QG model.

Graph2seq +RL+ BERT (Chen et al., 2020): is a BERT enhanced Graph2seq QG model with reinforcement learning.

QQP & QAP with BERT (Zhang and Bansal, 2019): combines the QG task and QA task with BERT.

Syn-QG (Dhole and Manning, 2020): is a rule-based QG model that uses the PropBank argument descriptions and VerbNet state predicates to incorporate shallow semantic content. It is a SOTA model in sentence-level QG task.

Recurrent BERT (Chan and Fan, 2020): employs the pre-trained BERT language model to tackle question generation tasks. It is also a SOTA model in sentence-level QG task.

We evaluate the performance of our models using BLEU (Papineni et al., 2002) and ROUGE-L (Lin, 2004), which are widely used in previous QG works.

3.3 Implementation Details

We fix the 300-dim GloVe vectors for the most frequent 70,000 words in the training set. We compute the 1024-dim BERT embeddings on the fly for each word in text using a trainable weighted sum of all BERT layer outputs. The embedding sizes of the case, answer, copy, POS , and NER tags are set of 3, 3, 3, 12 and 8, respectively. We set the hidden state size of BiLSTM to 150 so that the concatenated state size for both directions is 300. The size of all other hidden layers is set to 300. We apply a variational dropout rate of 0.4 after word embedding layers and 0.3 after RNN layers. The number of GNN hops in both encoder and decoder is set to 4. We use Adam (Kingma and Ba, 2014) as the optimizer and the learning rate is set to 0.001. We reduce the learning rate by a factor of 0.5 if the validation BLEU-4 score stops improving for three epochs. We stop the training when no improvement is seen for 10 epochs. We clip the gradient at length 10. The batch size is set to 60 for both SQuAD and MARCO. The beam search width is set to 5. All hyperparameters are tuned on the development set.

[1] https://res.qyzhou.me/redistribute.zip
[2] https://microsoft.github.io/msmarco/

Models	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-L
NQG++ (Zhou et al., 2018)	42.46	26.33	18.46	13.51	-
MPQG (Song et al., 2018)	-	-	-	14.71	42.60
Answer-focused Position-aware model (Sun et al., 2018)	43.02	28.14	20.51	15.64	-
s2sa-at-mp-gsa (Zhao et al., 2018)	44.51	29.07	21.06	15.82	44.24
ASs2s (Kim et al., 2019)	-	-	-	16.17	-
To the Point Context (Li et al., 2019)	44.40	29.48	21.54	16.37	44.73
QQP & QAP with BERT (Zhang and Bansal, 2019)	-	-	-	18.65	46.76
QG-pg (Jia et al., 2020)	43.63	29.21	21.79	16.93	-
Graph2seq + RL + BERT (Chen et al., 2020)	-	-	-	18.30	45.98
Syn-QG (Dhole and Manning, 2020)	45.55	30.24	23.84	18.72	-
Recurrent BERT (Chan and Fan, 2020)	48.29	33.12	24.78	19.14	47.07
Our Model	**50.82**	**34.73**	**25.64**	**20.33**	**48.94**

Table 1: Automatic evaluation results on the **sentence-level SQuAD** test set shared by (Zhou et al., 2018)

Models	BLEU-4
s2sa-at-mp-gsa (Zhao et al., 2018)	16.02
Answer-focused Position-aware model (Sun et al., 2018)	19.45
QG with semantic matching (Ma et al., 2020)	20.46
QG-pg (Jia et al., 2020)	21.61
Graph2seq +RL+ BERT (Chen et al., 2020)	22.59
Our Model	**23.87**

Table 2: Automatic evaluation results on the **sentence-level MARCO** test set

4 Results and Analysis

4.1 Main Results

Table 1 shows the experimental results of the SQuAD sentence-level dateset. For fair comparison, we report the results on sentence-level dataset excludes paragraph-level results (Dong et al., 2019; Bao et al., 2020; Qi et al., 2020).

In terms of BLEU-4 regarded as the main evaluation metric for text generation, our model yields the best results, with 20.33. We achieve state-of-the-art results on SQuAD for sentence-level QG.

We perform experiments on MARCO and achieve the state-of-the-art results as shown in Table 2. SQuAD and MARCO are built in different ways. The questions in SQuAD are generated by crowd-workers. However, questions in MARCO are sampled from real user queries. The experimental results on two datasets validate the generalization and robustness of our models.

4.2 Human Evaluation

To further assess the quality of generated questions, we perform a human evaluation to compare our model with the strong baseline of **Graph2seq +RL+ BERT** (Chen et al., 2020). We randomly select 100 samples from SQuAD and ask three annotators to score these generated questions according to three aspects:

Fluency: measures whether a question is grammatical and fluent;

Relevancy: measures whether the question is relevant to the input context;

Answerability: indicates whether the question can be answered by the given answer.

The rating score is set to [0, 5]. The evaluation results are shown in Table 3. Our model receives higher scores on all three metrics, indicating that our generated questions have higher quality in different aspects.

Models	Fluency	Relavancy	Answerability
baseline	3.81	3.69	3.74
our model	**4.24**	**4.33**	**4.26**
ground-truth	4.89	4.38	4.75

Table 3: Human evaluation results.

4.3 Ablation Study

As shown in Table 4, we perform an ablation study to systematically assess the impact of different model components (BERT, relational embedding, IGND) on the SQuAD test-set.

We remove the relational embedding in the encoder, the BLEU-4 score of the original model drops from 20.33 to 19.43, which indicates the importance of relational embedding. This is verified by comparing the performance of original w/o IGND model (19.01 BLEU-4 score) and other Graph2seq model to use the syntactic information baseline such as Graph2seq + RL + BERT (18.30 BLEU-4 score).

In addition, we remove the IGND and use the normal attention-based mechanism. The BLEU-4 score drops from 20.33 to 19.01 as shown in Table 4. From this result, the IGND improves the model performance. Moreover, the time cost of

the original model is only **1.1 times** more than the original w/o IGND model . In comparison to the original model that scores 20.33, the original w/o relational embedding and IGND drop significantly (almost 2 in BLEU-4 score).

We find that the pre-trained BERT embedding considerable impact the performance, and fine-tuning BERT embedding improves the performance, and demonstrates the power of large-scale pre-trained language models.

Models	BLEU-4
Original	**20.33**
- w/o BERT	19.37
- w/o relational embedding	19.43
- w/o IGND	19.01
- w/o relational embedding and IGND	18.42

Table 4: Ablation study on the SQuAD test set

4.4 Analysis of the Impact of Syntactic Information in Encoder

To understand the impact of the syntactic information, we calculated the words in the document that occurred in the ground truth question and their total attention score at the end of input attention layer as revealed in Figure 3.

We compare three models: **NQG++** (Zhou et al., 2018), which ignore the syntactic information; **Graph2Seq + RL + BERT** (Chen et al., 2020), which use the syntactic structure information but ignore the dependency relations. Our model uses syntactic information that includes both structure information and dependency relations. In our model, the average attention score is the highest, which indicates that syntactic information can improve the performance of encoder.

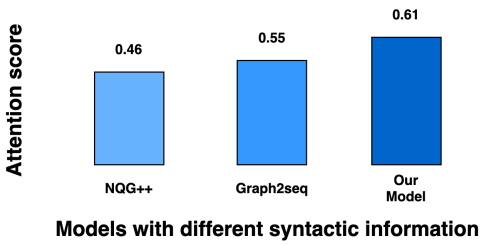

Figure 3: Average total attention score of words in the document that occurred in the ground truth question when using different model (SQuAD).

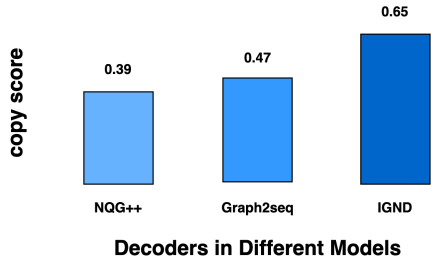

Figure 4: Average total copy probability score of words in the the ground truth question when using different model (SQuAD).

4.5 Analysis of the impact of IGND

To understand the impact of the IGND, we calculated for the words copied from the passage that occurred in the ground truth question. Their probability score at the decoding step, as revealed in Figure 4.

We compare with **NQG++** (Zhou et al., 2018) and **Graph2Seq + RL + BERT** (Chen et al., 2020). Our model with IGND has the highest average copy probability score, demonstrating that the IGND can improve the performance of copy mode by modeling previous information.

4.6 Sensitivity Analysis of Hyperparameters

We perform experiments on the Original model on the SQuAD to study the effect of the number of GNN hops. Figure 5 shows that our model is insensitive to the GNN hops and can achieve reasonably good results with various hops.

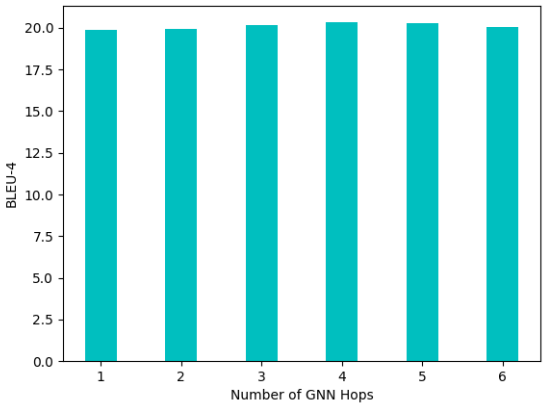

Figure 5: Effect of the number of GNN hops.

4.7 Case Study

We present some examples of questions generated by our model. Furthermore, we present a pair of

Sentence: donald davies at the national physical laboratory, independently developed the same message routing methodology as developed by baran.
Reference: what did donald davies develop ?
NQG++: who is donald davies ?
Graph2seq + RL + BERT: what did develop by baran?
Our Model: what did donald davies develop ?
Sentence: over time, wealth condensation can significantly contribute to the persistence of in equality within society
Reference: what has the highest impact on wealth accumulation and the resulting income inequality ?
NQG++: What is the persistence of inequality ?
Graph2seq + RL + BERT: what can contribute to the equality within society ?
Our Model: what can contribute to the persistence of inequality within society ?

Table 5: We show a few examples that illustrate the quality of generated text given a passage under different models. As we can see, incorporating dependency relations information helps the model identify which word is relevant to answer and thus makes the generated questions more relevant and specific.

examples, which have the same input sentence as shown in Table 5. We find that the question generated by **NQG++** and **Graph2Seq+RL+BERT** do not have the correct semantics, which copy the wrong word from the passage and can not find the right word as shown by **Graph2seq + RL + BERT** in Example 2. In contrast, our model can generate more answer-relevant questions than **NQG++** and **Graph2Seq+RL+BERT** baseline.

5 Related Work

Early works on QG (Mostow and Chen, 2009; Heilman and Smith, 2010) focused on the rule-based approaches that rely on heuristic rules or handcrafted templates, with low generalizability and scalability. Recent works adopted the attention-based sequence-to-sequence neural model for QG tasks, taking answer sentence as input and output the question (Du et al., 2017), which proved to be better than rule-based methods. (Zhou et al., 2018) proposed the feature-enriched encoder to encode the input sentence by concatenating word embedding with lexical features as the encoder input, and answer position are to inform the model to locate the answer. To generate a question for a given answer, (Sun et al., 2018; Kim et al., 2019; Song et al., 2018) applied various techniques to encode answer location information into an annotation vector corresponding to the word positions, thus allowing for better quality answer focused questions. (Liu et al., 2019; Chen et al., 2020) presented a syntactic features based method to represent words in the document and to decide what words to focus on while generating the question. (Chen et al., 2020) combined supervised and reinforcement learning in the training to maximize rewards that measure question quality. Furthermore, recent concurrent work applied the large-scale language model pre-training strategy for QG to achieve a new state-of-the-art performance (Chan and Fan, 2020).

Most existing QG approaches are unable to explicitly model the previously generated words. However, we perceive that previous generated words serve as auxiliary information in subsequent generation.

6 Conclusion

In this study, we designed the IGND for QG to alleviate the problem, which ignore the structure information and copied words in generated words at each decoding step. In addition, we proposed the relational graph encoder to capture the dependency relations information to improve the performance. For the sentence-level QG task on SQuAD and MARCO dataset, our method outperforms existing methods by a significant margin and achieves the new state-of-the-art results. Future directions include investigating more effective ways of utilizing previous generation information and exploiting Graph2Seq models with GNN-based decoder for question generation from structured data like knowledge graphs or tables.

7 Acknowledgments

The authors wish to thank the anonymous reviewers for their helpful comments. This work was partially funded by China National Key RD Program (No. 2017YFB1002104), National Natural Science Foundation of China (No. 61976056, 62076069), Shanghai Municipal Science and Technology Major Project (No.2021SHZDZX0103).

References

Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Jianfeng Gao, Songhao Piao, Ming Zhou, et al. 2020. Unilmv2: Pseudo-masked language models for unified language model pre-training. In *International Conference on Machine Learning*, pages 642–652. PMLR.

Ying-Hong Chan and Yao-Chung Fan. 2020. A recurrent BERT-based model for question generation. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 154–162, Hong Kong, China. Association for Computational Linguistics.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. pages 1870–1879.

Yu Chen, Lingfei Wu, and Mohammed J. Zaki. 2020. Reinforcement learning based graph-to-sequence model for natural question generation. In *Proceedings of the 8th International Conference on Learning Representations*.

Kyunghyun Cho, Bart van Merriënboer Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation.

Kaustubh Dhole and Christopher D. Manning. 2020. Syn-QG: Syntactic and shallow semantic rules for question generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 752–765, Online. Association for Computational Linguistics.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems*, pages 13063–13075.

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

Michael Heilman and Noah A. Smith. 2010. Good question! statistical ranking for question generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 609–617, Los Angeles, California. Association for Computational Linguistics.

Xin Jia, Wenjie Zhou, Xu Sun, and Yunfang Wu. 2020. How to ask good questions? try to leverage paraphrases. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6130–6140, Online. Association for Computational Linguistics.

Yanghoon Kim, Hwanhee Lee, Joongbo Shin, and Kyomin Jung. 2019. Improving neural question generation using answer separation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6602–6609.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jingjing Li, Yifan Gao, Lidong Bing, Irwin King, and Michael R. Lyu. 2019. Improving question generation with to the point context. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3216–3226, Hong Kong, China. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Bang Liu, Mingjun Zhao, Di Niu, Kunfeng Lai, Yancheng He, Haojie Wei, and Yu Xu. 2019. Learning to generate questions by learningwhat not to generate. New York, NY, USA. Association for Computing Machinery.

Xiyao Ma, Qile Zhu, Yanlin Zhou, and Xiaolin Li. 2020. Improving question generation with sentence-level semantic matching and answer position inferring. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:8464–8471.

Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Margaret Mitchell, Xiaodong He, and Lucy Vanderwende. 2016. Generating natural questions about an image. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1802–1813, Berlin, Germany. Association for Computational Linguistics.

Jack Mostow and Wei Chen. 2009. Generating instruction automatically for the reading strategy of self-questioning.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. *choice*, 2640:660.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA. Association for Computational Linguistics.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. ProphetNet: Predicting future n-gram

for sequence-to-SequencePre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2401–2410, Online. Association for Computational Linguistics.

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

Linfeng Song, Zhiguo Wang, Wael Hamza, Yue Zhang, and Daniel Gildea. 2018. Leveraging context information for natural question generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 569–574, New Orleans, Louisiana. Association for Computational Linguistics.

Xingwu Sun, Jing Liu, Yajuan Lyu, Wei He, Yanjun Ma, and Shi Wang. 2018. Answer-focused and position-aware neural question generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3930–3939, Brussels, Belgium. Association for Computational Linguistics.

Duyu Tang, Nan Duan, Tao Qin, and Ming Zhou. 2017. Question answering and question generation as dual tasks.

Xingdi Yuan, Tong Wang, Caglar Gulcehre, Alessandro Sordoni, Philip Bachman, Sandeep Subramanian, Saizheng Zhang, and Adam Trischler. 2017. Machine comprehension by text-to-text neural question generation.

Shiyue Zhang and Mohit Bansal. 2019. Addressing semantic drift in question generation for semi-supervised question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2495–2509, Hong Kong, China. Association for Computational Linguistics.

Yao Zhao, Xiaochuan Ni, Yuanyuan Ding, and Qifa Ke. 2018. Paragraph-level neural question generation with maxout pointer and gated self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3901–3910, Brussels, Belgium. Association for Computational Linguistics.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2018. *Neural Question Generation from Text: A Preliminary Study*, pages 662–671.