Entity Guided Question Generation with Contextual Structure and Sequence Information Capturing

Qingbao Huang,[1,2] Mingyi Fu,[2] Linzhang Mo,[2] Yi Cai,[1,3] *
Jingyun Xu,[1] Pijian Li,[2] Qing Li,[4] Ho-fung Leung[5]

[1]School of Software Engineering, South China University of Technology, Guangzhou, China
[2]School of Electrical Engineering, Guangxi University, Nanning, China
[3]Key Laboratory of Big Data and Intelligent Robot (SCUT), MOE of China
[4]Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China
[5]Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, China
qbhuang@gxu.edu.cn, {1812392009, 1912302009, 1912302005}@st.gxu.edu.cn, ycai@scut.edu.cn

Abstract

Question generation is a challenging task and has attracted widespread attention in recent years. Although previous studies have made great progress, there are still two main shortcomings: First, previous work did not simultaneously capture the sequence information and structure information hidden in the context, which results in poor results of the generated questions. Second, the generated questions cannot be answered by the given context. To tackle these issues, we propose an entity guided question generation model with contextual structure information and sequence information capturing. We use a Graph Convolutional Network and a Bidirectional Long Short Term Memory Network to capture the structure information and sequence information of the context, simultaneously. In addition, to improve the answerability of the generated questions, we use an entity-guided approach to obtain question type from the answer, and jointly encode the answer and question type. Both automatic and manual metrics show that our model can generate comparable questions with state-of-the-art models. Our code is available at https://github.com/VISLANG-Lab/EGSS.

Introduction

The Question Generation (QG) task is defined as the generation of a sentence-related, answerable, and grammatically correct question from a given passage, which is widely applied in the dialogue system (Wang et al. 2018), question answering (Tang et al. 2017), machine reading comprehension (Yuan et al. 2017), and automatic tutoring systems (Danon et al. 2017).

Traditional QG methods rely on manually curated rules or templates (Heilman et al. 2009; Hussein et al. 2014), which usually suffer from time-consuming, low generalizability and low scalability. Benefiting from the success of the encoder-decoder structure in machine translation (Bahdanau et al. 2015), Du et al. (2017) proposed a sequence-to-sequence model with attention mechanism for QG and achieved promising results. Since then, a surge of follow-up enhanced models (Zhou et al. 2017; Song et al. 2018a; Kumar et al. 2018a) have been proposed, most of them purely leverage Recurrent Neural Networks (RNNs) to capture sequence information of the context to generate questions. However, these sequence-information-aware models ignore rich structure information (e.g., syntactic dependency relations) hidden in the context.

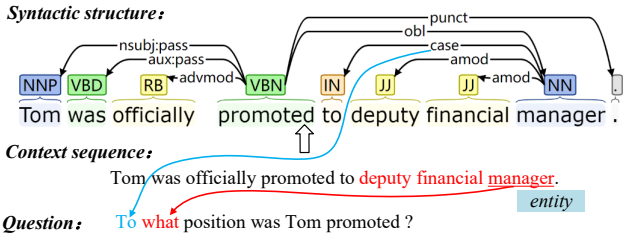

Figure 1: An example from SQuAD (Simplified for the sake of description). The phrase in red is the target answer. Syntactic structure information (obtained by Universal Stanford dependencies tool), sequence information of the context, and entities in the answer will boost question generation.

Dependency relations, especially long distance syntactic relations, are very useful for understanding complex sentence structures (e.g., long clauses or complex scoping). For the strong ability of aggregation, Graph Neural Networks (GNN) are applied to capture the dependency structure of the input passage to generate questions (Chen et al. 2020; Chai et al. 2020). Chen et al. (2020) applied a Bidirectional Gated GNN to capture the structure information between words in a sentence. Chai et al. (2020) used GNNs to encode the context and answer, respectively, to obtain the fused representations. Pan et al. (2020) used semantic information to construct an attention-based Gated Graph Neural Networks for generating deep questions. All of these models have shown strong abilities to capture structure information.

Although these graph-based studies have made marked progress in QG, a significant gap on performance still re-

*Corresponding author: Yi Cai (ycai@scut.edu.cn)

mains between machines and human. We consider that probable causes are twofold: First, using alone graphs pruned by DP is insufficient to convey the total information of the passage, namely, sequence information in the passage is not captured by the graph. Second, the DP, especially obtained by external tools, may bring error propagation problem.

Taking the aforementioned two classes of approaches into overall consideration, we argue that the structure information and sequence information of the context should be captured simultaneously, while the absence of either one may affect the quality of the generation question. As shown in Figure 1, the dependency relation $case$ between "to" and "manager" will facilitate the generation of the leading words "To what" in the question sentence.

Furthermore, some generated questions cannot be answered by the context (e.g. "Why was Tom promoted to a deputy position?"), which is fatal to QG and also means that the generated questions are insufficient in answerability. We deem that the answerability of a question is mainly influenced by the interrogative words and the relevance between the question and the answer. Previous studies mainly applied neural networks to predict the types of questions from the answers (Zhou et al. 2019; Wu et al. 2020), or used the predicted types of questions combined with the position information of the answers to improve the accuracy of generating interrogative words and the relevance between the questions and the answers (Hu et al. 2018; Zi et al. 2019; Chen et al. 2019). However, these approaches are implicit and difficult to ensure the types of questions predicted are correct. Therefore, we consider introducing entity-guided and rule-based approach which will be helpful to more accurately generate interrogative word. In Figure 1, the entity "manager" can help to generate the interrogative "what".

Inspired by above, we propose an Entity-Guided dual-channel encoding framework to capture contextual Structure information and Sequence information (EGSS) simultaneously. Specifically, we introduce DP to build syntactic relations and Graph Convolutional Networks (GCNs) over it to aggregate adjacent nodes and update the representation of structure information. Meanwhile, we employ a Bidirectional Long Short Term Memory Network (Bi-LSTM) to capture the contextual sequence information. We propose an entity guided method to guarantee a strong correlation between an answer and the generated interrogative word. First, we identify the answer by a named entity. Then, each entity is mapped to a corresponding question type in a rule-based way. Finally, we encode both the answer and the question type to guide the question generation. The method models the answer and question type in an explicit way, and considers the interrogative word and the answer's relevance to the question, thus improving the answerability of the generated question.

We conclude the contributions as follows:

- We propose a dual-channel encoding question generation model based on GCN and Bi-LSTM to capture structure information and sequence information of the context.
- We propose an entity-guided method to guarantee a correlation between an answer and the generated interrogative word, which can markedly improve the accuracy of question types and the answerability of generated questions.
- Experiments show that our model outperforms existing stong baselines (except for the state-of-the-art pre-trained language models) by a significant margin on the widely-used SQuAD dataset. Human evaluations show that our model can generate questions with better grammaticality, relevancy, and answerability.

Related Work

In recent years, question generation has attracted the attention of researchers. Previous methods on this task can be roughly categorized into two lines: rule-based approaches and neural network-based approaches.

Rule-based techniques for QG usually rely on manually-designed rules or templates to transform a piece of given text to questions (Heilman, 2010; Chali and Hasan, 2012; Lindberg et al. 2013; Labutov et al. 2015; Dhole et al. 2020). This rule-based approach requires a lot of labor. These rules or templates have many limitations that make them difficult to generalize. Moreover, the generated questions lack diversity due to the limited number of rules or templates.

To make up for the deficiency of rule-based methods, Du et al. (2017) applied sequence-to-sequence model to the QG task for the first time and obtained surprising performance. After that, many studies have been done on the basis of Seq2Seq (Zhao et al. 2018b; Kim et al. 2019; Li et al. 2019a; Wang et al. 2020; Yu et al. 2020). To make the model learn more language features and answer-related information, token lexical features and answer position information are usually input into the model (Zhou et al. 2017; Song et al. 2018; Kim et al. 2019; Jia et al. 2020). Similar to other text generation tasks, many previous work on QG also employ pointer or copy mechanism to overcome the OOV problem (Du and Cardie, 2018; Zhang and Bansal, 2019). Some works improve the quality of the generated questions by predicting the type of question or directly encoding the answer information (Hu et al. 2018; Zhou et al. 2019; Zi et al. 2019; Chen et al. 2020; Wu et al. 2020).

In recent years, because graph neural networks (Kipf et al. 2016; Gilmer et al. 2017; Hamilton et al. 2017) have made great progress in representation learning, many researchers use it to capture the relationship between words in context (Fan et al. 2019; Huang et al. 2020). Some researchers have also begun to explore the role of GNNs in the QG task. Liu et al. (2019) employed a GCN onto a Dependency Parsing (DP) tree to predict potential clue words, then fed them together with other features into a RNN-based encoder to encode the context. Chen et al. (2020) proposed the graph-to-sequence model for QG. The authors applied a Bidirectional Gated GNN to capture the structure information between words in a sentence. Chai et al. (2020) used GNNs to interactive answer and passage to obtain new representations. To generate deep questions, Pan et al. (2020) employed an attention-based Gated GNNs model to encode semantic information of the context.

In addition, due to the continuous expansion of large-scale corpus, some pre-training models have also achieved

remarkable performance in QG tasks (Qi et al. 2020; Bao et al. 2020; Xiao et al. 2020).

Model

Overview

The task of question generation can be formulated as follows: given a sentence $X = x_1 x_2 \dots x_n$ (containing n words) with an answer $A = a_1 a_2 \dots a_m$ (containing m words), the purpose of QG is to generate a corresponding question $Y = y_1 y_2 \dots y_l$ (containing l words).

Different from the common encoder-decoder architecture of previous work, we use a GCN encoder, a multi-feature encoder, and an AT encoder respectively in the encoding stage (cf. Figure 2). Among them, the GCN encoder is used to capture the structure information of context, the multi-feature encoder is used to capture contextual sequence information, and the AT encoder is employed to guide interrogative word generation and enhance the relevance of the generated question to the answer.

GCN Encoder

In natural language processing, GCN can effectively capture the relationship between words, well represent the context information, and play a very important role in text generation. In this section, we use GCN to encode sentences and get their representation, and then we focus on graph construction and node representation update.

Graph Construction For a sentence $X = x_1 x_2 \dots x_n$, we treat each word x_k as a node and the feature of node is the corresponding word representation h_{sk}, where h_{sk} is obtained through Glove embedding (Pennington et al. 2014). For edges, we first consist a fully-connected undirected graph, then prune some irrelevant edges through the dependency parsing tree to obtain a sparse graph. A sentence graph G is constructed as follows:

$$G = (\mathcal{V}, \xi), \tag{1}$$

where $\mathcal{V}$ represents the set of nodes and ξ represents the set of edges that connect these nodes.

We represent the graph structure with $n \times n$ adjacency matrix Λ, the adjacency matrix represents the relationship between nodes in GCN, and its initial state is a fully connected graph. By removing the unrelated edges with DP, a sparse graphical structure is obtained. Specifically, we use the Universal Stanford dependencies tool (De Marneffe et al. 2014) to obtain the relationship between words in the sentence. If there is some relationship between words x_i and x_k, we set the corresponding element in the adjacency matrix to 1, otherwise it is 0.

Node Update Representation Given a target node i and a neighboring node $j \in \mathcal{N}(i)$ in a sentence, where $\mathcal{N}(i)$ is the set of nodes neighboring with node i. h_{si} and h_{sj} are respectively the representations of node i and node j. We further learn a fully connected layer over concatenated node features h_{si} and h_{sj} to obtain the correlation score between node i and j:

$$s_{ij} = w_0^T \sigma(W_0[h_{si}; h_{sj}] + b_0), \tag{2}$$

where w_0, W_0 and b_0 are learned parameters, σ is the nonlinear activation function, and $[h_{si}; h_{sj}]$ represents the concatenation operation of h_{si} and h_{sj}. To obtain the edge weight β_{ij} between node i and node j, softmax operation was performed on the correlation score s_{ij} to obtain it. The calculation formula is as follows:

$$\beta_{ij} = \frac{exp(s_{ij})}{\sum_{j \in \mathcal{N}(i)} exp(s_{ij})}. \tag{3}$$

Further, we update the h_{si} of $k+1$ layer by aggregating the information of the neighbor node j of node i and node i at k layer. This propagation is denoted as:

$$h_{si}^{(l+1)} = \sigma(h_{si}^{(l)} + \sum\nolimits_{j \in \mathcal{N}(i)} \Lambda_{ij} \beta_{ij} (W_1 h_{sj}^{(l)} + b_1)), \tag{4}$$

where σ denotes a non-linear function, W_1 and b_1 are learned parameters, Λ_{ij} represents an element in the adjacency matrix. Finally, the output H_s of a stacked l-layer GCN can be obtained according to the following formula:

$$H_s = \{h_{si}^{(l+1)}\}_{i=1}^{n}. \tag{5}$$

Multi-feature Encoder

The input of multi-feature encoder contains word embeddings, linguistic features, and answer tags. We use pretrained Glove embeddings (Pennington et al. 2014) to represent each word x_i^g. For linguistic features, we transform POS and NER tags into continuous representation z_i^p and z_i^n, respectively. Following (Zhou et al. 2017), we adopt a BIO label to indicate the relative position of the answer in the sentence. Then we get the embedding z_i^b of answer tags. The final embedding of the i-th word in the sentence is denoted as:

$$x_i^{fe} = [x_i^g; z_i^p; z_i^n; z_i^b], \tag{6}$$

where $[;]$ denotes the concatenation operation.

We employ a Bi-LSTM to encode the input $X^{fe} = \{x_i^{fe}\}_{i=1}^{n}$ to get a contextualized representation for each token:

$$h_{ri} = BiLSTM(h_{r(i-1)}, x_i^{fe}), \tag{7}$$

where $h_{ri} = [\overrightarrow{h_{ri}}; \overleftarrow{h_{ri}}]$ is the hidden states at the i-th time step of Bi-LSTM. The contextualized representation of the sentence can be denoted as $H_r = (h_{r1}, h_{r2}, \dots, h_{rn})$.

Finally, we use a concatenation operation to fuse outputs of GCN-Encoder and multi-feature Encoder, and get the final contextualized representation $H_d = [H_s; H_r]$.

AT Encoder

Inspired by the question-type driven question generation proposed by Zhou et al. (2019), we propose an Entity Guided Interrogative Generation method to initialize the decoder with the answer information and question types to improve the answerability of the generated questions. Different from Zhou's approach that utilizes the hidden layer vector of the answer position to predict the type of question, we obtained the type of the question through the rule-based method, which was more accurate in classifying the question. More specifically, we first perform NER on the answer.

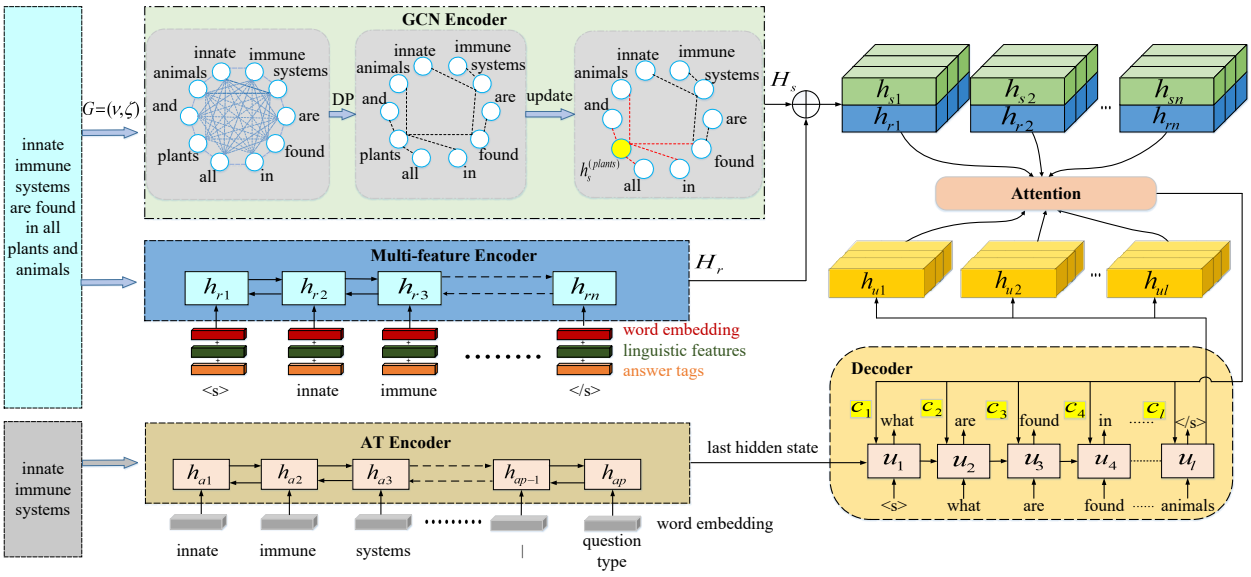

Figure 2: Illustration of our proposed EGSS model for QG task. A GCN encoder is used to capture contextual structure information, a Bi-LSTM encoder is used to capture contextual sequence information, and an AT encoder is used to encode question types and answers to guide the generation of interrogative words and improve the relevance of the question to the answer. In addition, DP represents dependency parsing, update denotes the update of the node.

Second, classify the question. If the answer is a person's name, place, time, and number, the corresponding question types are *who*, *where*, *when*, and *how many/much*, respectively. If the answer is a noun or a noun phrase, the question type is *what*. If the answer is a sequence word or an adjective comparative, then the question type is set to *which*. If the answer contains keywords such as "reason", "because", the question type is *why*. The settings other than these types are *other*.

We employ a Bi-LSTM as the AT encoder, and input the answer $A = a_1, a_2, \ldots, a_m$ and question types s into the AT encoder. The final input form A_s of the AT encoder is the answer and question type separated by separator $|$, A_s thus can be described as:

$$A_s = a_1, a_2, \ldots, a_m, |, s. \tag{8}$$

For the convenience of the following application, we further express A_s as:

$$A_s = a_{s1}, a_{s2}, \ldots, a_{sp}, \tag{9}$$

where a_{si} represents Glove embedding of the i-th token in A_s. Finally, last hidden state h_{ap} of the AT encoder is obtained by the following formula:

$$h_{ai} = BiLSTM(h_{a(i-1)}, a_{si}), \tag{10}$$

where $h_{ai} = [\overrightarrow{h_{ai}}; \overleftarrow{h_{ai}}]$ is the hidden states at the i-th time step (h_{ap} denotes the last hidden state).

Decoder

We employ an LSTM as the decoder to generate the question. The hidden state u_t of LSTM at time t is denoted as:

$$u_t = LSTM(u_{t-1}, y_{t-1}), \tag{11}$$

where u_{t-1} represents the hidden state of LSTM at time t-1, y_{t-1} represents the word generated by the decoder at time t-1. When t=0, $u_0 = h_{ap}$ stands for the decoder's initial state.

We employ the Luong et al. (2015)'s attention mechanism to obtain row attention scores r_t over the input sequence which has a vocabulary of V_s:

$$r_t = {H_d}^T W_2 u_t, \tag{12}$$

where H_d denotes contextualized representation of the sentence, W_2 stands for a learning matrix. Then, we obtain the context vector c_t and update the decoder state u_t to get $\widehat{u}$:

$$c_t = H_d softmax(r_t), \tag{13}$$

$$\widehat{u}_t = tanh(W_3[u_t; c_t]). \tag{14}$$

To solve the problem of rare words, the decoder applies a copy mechanism (See et al. 2017). Similar to (Zhao et al. 2018), we directly leverage row attention scores r_t as the score of the copy mechanism $score_1$. Further, we calculated the score for words generated from the high frequency vocabulary V_h:

$$score_2 = W^{V_h} \widehat{u}_t. \tag{15}$$

Finally, the probability distribution $p(y_t|\{y_{<t}\})_{final}$ of each word at t step can be calculated as follow:

$$p(y_t|\{y_{<t}\})_{final} = softmax([score_1; score_2]), \tag{16}$$

where the dimension of $[score_1; score_2]$ is $|V_h| + |V_s|$. If a word appears in both V_s and V_h, we sum up the probabilities of the two parts as its final probability.

Models	Split2						Split1					
	B1	B2	B3	B4	R-L	ME	B1	B2	B3	B4	R-L	ME
Seq2Seq (Du et al. 2017)	- -	- -	- -	- -	- -	- -	43.09	25.96	17.50	12.28	39.75	16.62
NQG++ (Zhou et al. 2017)	42.36	26.33	18.46	13.51	- -	- -	- -	- -	- -	- -	- -	- -
S2Sa-at-mp-gsa (Zhao et al. 2018)	44.51	29.07	21.06	15.82	44.24	19.67	43.47	28.23	20.40	15.32	43.91	19.29
NQG-Knowledge (Gupta et al. 2019)	42.78	26.68	18.48	13.30	41.22	19.15	- -	- -	- -	- -	- -	- -
ASs2s (Kim et al. 2019)	- -	- -	- -	16.17	43.96	19.92	- -	- -	- -	16.20	43.96	19.92
Unified-model (Zhou et al. 2019)	43.11	29.13	21.39	16.31	- -	-	- -	- -	- -	- -	- -	- -
CGC-QG (Liu et al. 2019)	46.58	30.9	22.82	17.55	44.53	21.24	- -	- -	- -	- -	- -	- -
CS2S-VR-A (Liu et al. 2020)	45.28	29.58	21.45	16.13	43.98	20.59	- -	- -	- -	- -	- -	- -
Multi-stage Att (Tuan et al. 2020)	46.60	31.94	23.44	17.76	45.89	21.56	45.13	30.44	23.40	17.09	45.81	21.25
Refine Net (Nema et al. 2019)	47.27	31.88	23.65	18.16	47.14	23.40	- -	- -	- -	- -	- -	- -
Ans-pivot-QG (Wang et al. 2020)	48.26	29.23	22.37	16.42	43.07	18.95	- -	- -	- -	- -	- -	- -
G2S$_s$-Bert-RL (Chen et al. 2020)*	- -	- -	- -	18.30	45.98	21.70	- -	- -	- -	17.94	46.02	21.76
Syn-QG (Dhole et al. 2020)*	45.55	30.24	23.84	18.72	- -	- -	- -	- -	- -	- -	- -	- -
EGSS w/o (Answer & QType)	49.53	32.58	23.86	18.01	45.27	22.38	49.20	32.57	23.72	17.99	45.23	21.88
EGSS (ours)	**50.86**	**34.42**	**25.44**	**19.45**	**47.29**	**23.54**	**50.11**	**33.26**	**24.20**	**18.47**	**47.04**	**21.93**

Table 1: Automatic evaluation on Split1 and Split2 of SQuAD. $*$ are dependency-based models, while other baselines are sequence-based. B1 is short for BLEU-1, B2 for BLEU-2, B3 for BLEU-3, B4 for BLEU-4, ME for METEOR, and R-L for ROUGE-L. We conduct experiments on their released codes or copy from their paper. In each column, we bold / underline the best performance over all / baseline methods, respectively. Our EGSS model significantly outperforms baselines with p $<$0.001.

Models	B4	R-L	ME
GPT2-ACS (Liu et al. 2020)	18.87	43.6	25.15
UniLMv2 (Bao et al. 2020)	26.29	53.22	27.16
Ernie-Gen (Xiao et al. 2020)	25.57	53.31	26.89
ProphnetNet (Qi et al. 2020)	**26.72**	**53.79**	**27.64**
EGSS (ours)	19.45	47.29	23.54

Table 2: Comparative experiments between EGSS and pre-trained language models on Split2.

Experiments

Dataset

We conduct experiments on the accessible part of widely-used SQuAD datasets (Rajpurkar et al. 2016). Specifically, we conduct sentence-level question generation experiments on Split1 (Du et al. 2017) and Split2 (Zhou et al. 2017), respectively. In Split1, the original dev set is used as test set, and the original training set is randomly divided into training set and dev set at a 9:1 ratio. In Split2, the SQuAD training set remains the same and the original dev set is randomly split into our dev set and test set with the ratio 1:1.

Evaluation Metrics

Automatic Evaluation Metric We conduct automatic evaluation with the following metrics:

BLEU (Papineni et al. 2002) measures precision by how much the words in predictions appear in reference sentences. BLEU-1 (B1), BLEU-2 (B2), BLEU-3 (B3), and BLEU-4 (B4) use 1-gram to 4-gram for calculation, respectively.

METEOR (ME) (Denkowski and Lavie, 2014) is based on the harmonic mean of unigram precision and recall, with recall weighted higher than precision.

ROUGE-L (R-L) (Lin, 2014) measures recall by how much the words in reference sentences appear in predictions using Longest Common Subsequence based statistics.

Human Evaluation Metric To further assess the quality of generated questions, we perform human evaluation to compare our model with Seq2Seq (Du et al. 2017), S2Sa-at-mp-gsa (Zhao et al. 2018), G2S-Bert-RL (Chen et al. 2020), and Syn-QG (Dhole et al. 2020).

We randomly select 100 samples from the generated questions and hire 5 evaluators to evaluate them according to three aspects: Grammaticality (GRAM) to measure the grammatical correctness and fluency of the generated questions, Relevancy (REL) to measure whether the generated question is relevant to the context, and Answerability (ANS) to assess whether the generated question can be answered by the context. For fair comparison, score 1/2/3/4/5 is applied to each metric during annotation following (Dhole et al. 2020). For each metric, we averaged the scores of the five evaluators as the final score.

Baselines

Seq2Seq: Du et al. (2017) proposed a seq2seq model with attention mechanism.

NQG++: Zhou et al. (2017) devised a seq2seq model with copy and attention mechanism, enhanced with answer position features and lexical features.

S2sa-at-mp-gsa: Zhao et al. (2018) used a gated attention encoder and a maxout pointer decoder for the QG task.

NQG-Knowledge: Gupta et al. (2019) used external knowledge to enhance question generation models.

ASs2s: Kim et al. (2019) proposed an answer-separated

Seq2Seq model by replacing the answer in the input sequence with some specific words.

Unified-model: Zhou et al. (2019) proposed a question-type driven question generation model that used neural networks to predict question types from answers.

CGC-QG: Liu et al. (2019) proposed a multi-task learning framework to guide the model to learn the accurate boundaries between copying and generation.

G2Ss-Bert-RL: Chen et al. (2020) proposed a RL based Graph2Seq model for QG.

Refine Net: Nema et al. (2019) leveraged a Preliminary Decoder and a Refinement Decoder with Dual Attention Network to generate questions.

Multi-stage Att: Tuan et al. (2020) represented the relevant context via a multi-stage attention mechanism to incorporate interactions across multiple sentences.

Syn-QG: Dhole et al. (2020) proposed a QG method based on syntactic and shallow semantic rules.

CS2S-VR-A: Liu et al. (2020) proposed an ACS-aware QG model. For fair comparison, we just introduce its CS2S-VR-A version, which only input passages and answers.

Experiments Settings

We utilize pre-trained Glove word embeddings, which dimension is set to 300. We set the dimension of Answer tags, POS tags, and NER tags with 100, 60, and 50, respectively. The hidden size of LSTM are 300 for all encoder and decoder. For the GCN encoder, the level of stacked layer is 4. We make the vocabulary with top 45000 frequency words. When training, we use SGD optimizer with 0.2 dropout rate, 0.01 learning rate and train our model for 30 epochs. For the decoder part, we use beam search with beam size of 12 to get the final result.

Results and Analysis

Automatic Evaluation Table 1 shows experimental results of our EGSS model on SQuAD.

For Split2, we achieve a strong improvement in BLEU, R-L and ME. It can be seen that our model significantly outperforms other baselines on B1, B2 and B3, respectively. In terms of B4 that is often regarded as the main evaluation metric for QG, our model outperforms the previous best result by 0.73% which are a large margin for this challenging task. Compared to Refine Net the best previous model on the R-L and ME of Split2, our model obtains improvement of 0.15% and 0.14%, respectively. We have done significant test comparing our model with NQG++, S2Sa-at-mp-gsa, Unified-model, Refine Net and G2S-Bert-RL. We run all these models ten times. The results shows that our model significantly outperforms them with all p-values <0.001. In general, the experimental results of our model are better than those that only rely on textual structure information or sequence information, which proves the validity of our model.

We also conduct experiments follow the split of Du et al. (2017). Similarly, our model achieves good performance in Split1 on each metric. Specially, our model outperforms the previous best result (i.e., G2S-Bert-RL) on B4, R-L, and ME, by 0.53%, 1.02%, and 0.17%, respectively. Our full model removing AT encoder module (i.e., Ans and Qtype) also achieves a comparable performance with baselines.

We further compare our EGSS with pre-trained language models (cf. Table 2). Although there is a gap between EGSS and the state-of-the-art pre-trained models, our model exceeds GPT2-ACS on B4 and R-L metrics. In general, our model achieves comparable results at a lower cost without the need for large amounts of external data and computing resources.

In addition, we compare our model with two question type-driven QG models on the accuracy of interrogative words (cf. Table 3). It can be seen that our model improved by 1.12% over the Unified model and by 29.18% over NQG++. It shows that the entity-guided and rule-based interrogative words generation approach is more effective than pure neural network-based methods. It also shows that injecting external symbolic representations could be helpful for neural language generation models.

Manual Evaluation We conducted human evaluations to analyze the quality of the question generated by EGSS. As shown in Table 4, our EGSS model outperforms the strong baseline models on all metrics. Specifically, our model achieves improvements of 3% on ANS (4.01 vs 3.89), 1% on REL (4.37 vs 4.33), and on GRAM (4.06 vs 3.93). Strong baselines (such as G2S-Bert-RL, Syn-QG) and our model both use DP. The evaluation results show that DP is beneficial to improve grammaticality of the generated question. Furthermore, our model achieve the best GRAM score, indicating that our GCN encoder is useful for capturing rich structure information hidden in the text. From Table 4, we also find that our model is better than the other baselines in terms of answerability, suggesting that the introduction of question types is beneficial for answerability. We will show some cases and analyze them detailedly in case study.

Ablation Study To investigate the effects of the GCN encoder module, the DP module, and the AT module in the EGSS model, we perform ablation studies. The results are shown in Table 5. It shows that the answer information provides certain performance improvement by 0.34% on B4. The question type information provides improvement by 1.1% on B4, 1.72% on R-L, 0.88% on ME. Compared to answer information, the question type information shows more important. We also remove answer and question type information together, namely removing the AT encoder part, the model produce massive drops by 1.44% on B4, 2.02% on R-L, 1.16% on ME. It can be seen that the AT encoder is one of the most important part in model.

Furthermore, we compare the influence of DP, GCN and Bi-LSTM parts to model in Table 5. When our model has only one vanilla GCN encoder, the scores of all metrics are relatively low. The scores of B4, R-L, and Meteor are 7.59, 37.23, and 15.12, respectively. Similarly, when our model has only one Bi-LSTM encoder, the performances are relatively low. Such modification only achieves 13.32 on B4, 39.66 on R-L, 17.39 on ME. However, we combine GCN and Bi-LSTM as encoders, the performance of our model is improved to 16.71 on B4, 44.43 on R-L and 21.51 on ME. Besides, the part of DP provides further improved by

Models	bqwa on split2
NQG++ (Zhou et al. 2017)	46.35%
Unified-model (Zhou et al. 2019)	74.41%
EGSS (ours)	**75.53**%

Table 3: Experiments on the beginning interrogative word accuracy (bqwa).

Models	GRAM	REL	ANS
Seq2Seq (Du et al. 2017)	3.48	3.28	3.18
S2Sa-at-mp-gsa (Zhao et al. 2018)	3.71	3.77	3.32
Unified-model (Zhou et al. 2019)	3.65	4.29	3.89
G2S-Bert-RL (Chen et al. 2020)	3.86	4.33	3.78
Syn-QG (Dhole et al. 2020)	3.93	4.28	3.85
EGSS (ours)	**4.06**	**4.37**	**4.01**

Table 4: Human evaluation results on the Split2 of SQuAD. GRAM is short for Grammaticality, REL for Relevancy, and ANS for Answerability.

1.3% on B4, 0.84% on R-L and 0.87% on ME. It can be concluded that using GCN and Bi-LSTM to simultaneously capture structure information and sequence information of context, respectively, is conducive to improving the performance of the model.

Case Study We present some examples of generated questions in Table 6. In Case 1, the Seq2Seq model lacks the ability to capture contextual structure information, leading to its failure to learn that "it" refers to "St. Johns River". The G2S-Bert-RL model does not make full use of the sequence information between words, resulting in the generated questions inconsistent with the context facts and unable to be answered by the given answer "Jean ribault". As for Refine Net, although it generates question without grammatical errors, it cannot be answered by given answers. In contrast, the questions generated by our model are more specific, and with better answerability and grammaticality.

In Case 2, Seq2Seq and G2S-Bert-RL do not make full use of the structure information and sequence information of the context, leading to the generation of low-quality questions, and making the generated questions poor in answerability. The question generated by Refine Net is better than Seq2Seq and G2S-Bert-RL in terms of grammar and answerability. But the question generated by our model is much more specific, which is exactly what high-quality questions require. In addition, we found that the Unified model did not correctly predict the interrogative words by the proposed implicit neural network predicting method.

The cases show that our model has ability to capture contextual structure and sequence information, thereby making the quality of the generated questions better. Moreover, the method of entity guiding question generation can improve the accuracy of interrogative word generation.

Models	B1	B2	B3	B4	R-L	ME
Our full model	**50.86**	**34.42**	**25.44**	**19.45**	**47.29**	**23.54**
- ans	50.57	34.08	25.06	19.11	46.99	23.26
- ans, qtype	49.53	32.58	23.86	18.01	45.27	22.38
- ans, qtype, dp	48.08	31.11	22.42	16.71	44.43	21.51
- ans, qtype, dp, gcn	43.27	25.73	18.45	13.32	39.66	17.39
- ans, qtype, dp, lstm	41.01	21.10	12.38	7.59	37.23	15.12

Table 5: Ablation studies on the Split2 of SQuAD. "-" means removing the corresponding modules.

Case 1	
Context:	French huguenot explorer **Jean ribault** charted the st. johns river in 1562 calling it the river of May because he discovered it in may.
Seq2Seq:	What created it in May?
Unified-model:	What discovered French huguenot explorer?
G2S-Bert-RL:	Who discovered Jean ribault?
Refine Net:	When did Jean ribault discovered johns river?
Ours:	Who charted the st. johns river in 1562?
Ground truth:	Who mapped the st. johns river in 1562?
Case 2	
Context:	**As a result of the american revolution**, john wesley was compelled in 1784 to break with standard practice and ordain two of his lay preachers as presbyters, thomas vasey and richard whatcoat.
Seq2Seq:	What was the name of the american revolution?
Unified-model:	What was wesley compelled to?
G2S-Bert-RL:	Why was thomas vasey and richard whatcoat break with standard practice?
Refine Net:	Why was wesley compelled to break with standard practice?
Ours:	Why was john wesley compelled in 1784 to break with standard practice and ordain two of his lay preachers as presbyters?
Ground truth:	Why was wesley compelled to break with standard practice and ordain two of his lay preachers as presbyters?

Table 6: Generated questions from different models. In all cases, the underlined part of the context is the given answer.

Conclusion

In this paper, we propose an Entity-Guided dual-channel encoding framework for QG. Our model uses a GCN encoder and a Multi-feature encoder to capture the structure information and sequence information of the context, respectively, to improve the quality of generated questions. Further, to improve the accuracy of interrogative word generation, we employ an AT encoder to guide interrogative word generation. Experimental results on SQuAD also show the effectiveness of our proposed method. In the future, we will explore the capture of structure information and sequence information of multiple sentences or paragraphs for QG and explore the generation of more inferential questions.

Acknowledgments

We thank the anonymous reviewers for valuable comments and thoughtful suggestions.

This work was supported by National Natural Science Foundation of China (62076100, 51767005), the National Key Research and Development Program of China, and the collaborative research grants from the Fundamental Research Funds for the Central Universities, SCUT (No. D2182480), the Science and Technology Planning Project of Guangdong Province (No.2017B050506004), the Science and Technology Programs of Guangzhou (No.201704030076, 201707010223, 201802010027, 201902010046), and the Hong Kong Research Grants Council, China (project no. PolyU1121417 and project no. C1031-18G), and an internal research grant from the Hong Kong Polytechnic University, China (project 1.9B0V).

References

Tuan, L. A.; Shah, D. J.; and Barzilay, R. 2020. Capturing Greater Context for Question Generation. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence,* 9065–9072.

Wang, B.; Wang, X.; Tao, T.; Zhang, Q.; and Xu, J. 2020. Neural Question Generation with Answer Pivot. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence,* 9138-9145.

Tang, D.; Duan, N.; Qin, T.; Yan, Z.; and Zhou, M. 2017. Question answering and question generation as dual tasks. In *arXiv preprint arXiv:1706.02027*.

Yuan, X.; Wang, T.; Caglar, G.; Sordoni, A.; Bachman, P.; Subramanian, S.; Zhang, S.; and Trischler, A. 2017. Machine comprehension by text-to-text neural question generation. In *Proceedings of the 2nd Workshop on Representation Learning for NLP,* 15-25.

Wang, Y.; Liu, C.; Huang, M.; and Nie, L. 2018. Learning to ask questions in open-domain conversational systems with typed decoders. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2193-2203.

Heilman, M.; and Smith, N. A. 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,* 609–617.

Danon, G.; and Last, M. 2017. A syntactic approach to domain-specific automatic question generation. In *arXiv:1712.09827, 2017*.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations* .

Sun, X.; Liu, J.; Lyu, Y.; He, W.; Ma, Y.; and Wang, S.; 2018. Answer-focused and Position-aware Neural Question Generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* , 3930-3939.

Du, X.; Shao, J.; and Cardie, C. 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),* volume 1, 1342–1352.

Zhou, Q.; Yang, N.; Wei, F.; Tan, C.; Bao, H.; and Zhou, M. 2017. Neural question generation from text: A preliminary study. In *In National CCF Conference on Natural Language Processing and Chinese Computing,* 662–671. Springer.

Hu, W.; Liu, B.; Ma, J.; Zhao, D.; and Yan, R. 2018. Aspect-based Question Generation. In *6th International Conference on Learning Representations (Workshop)*.

Zi, K.; Sun, X.; Cao. Y.; Wang, S.; Feng, X.; Ma.; and Cao, C. 2019. Answer-Focused and Position-Aware Neural Network for Transfer Learning in Question Generation. In *Knowledge Science, Engineering and Management - 12th International Conference, KSEM*, 339-352. Springer.

Zhou, W.; Zhang, M.; and Wu, Y. 2019. Question-type Driven Question Generation. In *Proceedings of EMNLP-IJCNLP 2019,* 6031-6036.

Liu, B.; Wei, H.; Niu, D.; Chen, H.; and He, Y. 2020. Asking Questions the Human Way: Scalable Question-Answer Generation from Text Corpus. In *Proceedings of The Web Conference 2020,* 2032-2043.

Chen, Y.; Wu, L.; N.; and Mohammed, J. 2020. Reinforcement Learning Based Graph-to-Sequence Model for Natural Question Generation. In *8th International Conference on Learning Representations, ICLR 2020*.

Song, L.; Wang, Z.; Hamza, W.; Zhang, Y.; and Gildea, D. 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,* 569-574.

Kumar, V.; Boorla, K.; Meena, Y.; Ramakrishnan, G.; and Li, Y. 2018a. Automating reading comprehension by generating question and answer pairs. In *In Pacific-Asia Conference on Knowledge Discovery and Data Mining,* 335-348.

Chali, Y.; and Hasan, S. A. 2012. Towards automatic topical question generation. In *COLING 2012, 24th International Conference on Computational Linguistics,* 475-492.

Lindberg, D.; Popowich, F.; Nesbit, J.; and Winne, P. H. 2013. Generating natural language questions to support learning on-line. In *Proceedings of the 14th European Workshop on Natural Language Generation,* 105-114.

Labutov, I.; Basu, S.; and Vanderwende, L. 2015. Deep questions without deep understanding. In *In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing* volume 1, 889–898.

Dhole, K. D.; and Manning, C. D. 2020. Syn-QG: Syntactic and Shallow Semantic Rules for Question Generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics,* 752-765.

Zhao, Y.; Ni, X.; Ding, Y.; and Ke, Q. 2018b. Paragraph-level neural question generation with maxout pointer and gated self-attention networks. In *EMNLP,* 3901-3910.

Kim, Y.; Lee, H.; Shin, J.; and Jung, K. 2019. Improving neural question generation using answer separation. In *AAAI,* volume 33, 6602-6609.

Li, J.; Gao, Y.; Bing, L.; King, I.; and Lyu, M. R. 2019a. Improving question generation with to the point context. In *EMNLP-IJCNLP,* pages 3216–3226, Hong Kong, China.

Nema, P.; Mohankumar, A. K.; Khapra, M.; Srinivasan, B. V.; and Ravindran, B. 2019. Let's ask again: Refine network for automatic question generation. In *EMNLP-IJCNLP,* pages 3312-3321, Hong Kong, China.

Hussein, H.; Elmogy, M.; and Guirguis, S. 2014. Automatic english question generation system based on template driven scheme. In *IJCSI* 11(6):45.

Yu, J.; Liu, W.; Qiu, S.; Su, Q.; Wang, K.; Quan, X.; and Yin, J. 2020. Low-Resource Generation of Multi-hop Reasoning Questions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics,* 6729-6739.

Jia, X.; Zhou, W.; Sun, X.; and Wu, Y. 2020. How to Ask Good Questions? Try to Leverage Paraphrases. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics,* 6130-6140.

Du, X.; and Cardie, C. 2018. Harvesting Paragraph-level Question-Answer Pairs from Wikipedia. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics,* 1907-1917.

Zhang, S.; and Bansal, M. 2019. Addressing Semantic Drift in Question Generation for Semi-Supervised Question Answering. In *EMNLP-IJCNLP,* 2495-2509.

Kipf, T. N.; and Welling, M., W. 2016. Semi-supervised classification with graph convolutional networks. In *ICLR, Toulon, France, April 24-26*.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning,* 1263-1272.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems,* 1024-1034.

Liu, B.; Zhao, M.; Niu, D.; lai, K.; He, Y.; Wei, H.; and Xu, Y. 2019. Learning to generate questions by learning what not to generate. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17* 1106-1118.

Chai, Z.; and Wan, X. 2020. Learning to Ask More: Semi-Autoregressive Sequential Question Generation under Dual-Graph Interaction. In *ACL,* 225-237.

Pan, L.; Xie, Y.; Feng, Y.; Chua, T, S.; and Kan, M, Y. 2020. Semantic Graphs for Generating Deep Questions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics,* 1463-1475.

De Marneffe, M. C.; Dozat, T.; Silveira, N.; Haverinen, K.; Ginter, F.; Nivre, J.; and Manning, C. D. 2014. Universal stanford dependencies: A cross-linguistic typology. In *LREC*, volume 14, pages 4585–4592.

Luong, M. T.; Pham, H.; and Manning, C. D. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing,* 1412–1421.

See, A.; Liu, P. J.; and Manning, C. D. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics,* 1073-1083.

Rajpurkar, P.; Zhang, J.; Lopyrev, K.; and Liang, P. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing,* 2383-2392.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics,* 311-318.

Denkowski, M.; and Lavie, A. 2014. Meteor universal: language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation,* 376-380.

Lin, C. Y. 2004. Rouge: A package for automatic evaluation of summaries. In *Stan Szpakowicz Marie-Francine Moens, editor, Text Summarization Branches Out: Proceedings of the ACL-04 Workshop,* 74-81.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP,* 1532-1543.

Qi, W.; Yan, Y.; Gong, Y.; Liu, D.; Duan, N.; Chen, J.; Zhang, R.; and Zhou, M. 2020. Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training. In *EMNLP (Findings),* 2401-2410.

Bao, H.; Dong, L.; Wei, F.; Wang, W.; Yang, N.; Liu, X.; Wang, Y.; Piao, S.; Gao, J.; Zhou, M.; and Hon, H. W. 2020. Unilmv2: Pseudo-masked language models for unified language model pre-training. In *arXiv preprint arXiv:2002.12804.*

Xiao, D.; Zhang, H.; Li, Y.; Sun, Y.; Tian, H.; Wu, H.; and Wang, H. 2020. ERNIE-GEN: An Enhanced Multi-Flow Pre-training and Fine-tuning Framework for Natural Language Generation. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence,* 3997-4003.

Gupta, D.; Suleman, K.; Adada, M.; McNamara, A.; and Harris, J. 2019. Improving Neural Question Generation using World Knowledge. In *arXiv preprint arXiv:1909.03716.*

Fan, W.; Ma, Y.; Li, Q.; He, Y.; Zhao, E.; Tang, J.; and Yin, D. 2019. Graph neural networks for social recommendation. In *The World Wide Web Conference,* 417-426.

Huang, Q.; Wei, J.; Cai, Y.; Zheng, C.; Chen, J.; Leung, H. F.; and Li, Q. 2020. Aligned Dual Channel Graph Convolutional Network for Visual Question Answering. In *ACL,* 7166-7176.