Learning to Ask More: Semi-Autoregressive Sequential Question Generation under Dual-Graph Interaction

Zi Chai, Xiaojun Wan
Wangxuan Institue of Computer Technology, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
{chaizi, wanxiaojun}@pku.edu.cn

Abstract

Traditional Question Generation (TQG) aims to generate a question given an input passage and an answer. When there is a sequence of answers, we can perform Sequential Question Generation (SQG) to produce a series of interconnected questions. Since the frequently occurred information omission and coreference between questions, SQG is rather challenging. Prior works regarded SQG as a dialog generation task and recurrently produced each question. However, they suffered from problems caused by error cascades and could only capture limited context dependencies. To this end, we generate questions in a semi-autoregressive way. Our model divides questions into different groups and generates each group of them in parallel. During this process, it builds two graphs focusing on information from passages, answers respectively and performs dual-graph interaction to get information for generation. Besides, we design an answer-aware attention mechanism and the coarse-to-fine generation scenario. Experiments on our new dataset containing 81.9K questions show that our model substantially outperforms prior works.

1 Introduction

Question Generation (QG) aims to teach machines to ask human-like questions from a range of inputs such as natural language texts (Du et al., 2017), images (Mostafazadeh et al., 2016) and knowledge bases (Serban et al., 2016). In recent years, QG has received increasing attention due to its wide applications. Asking questions in dialog systems can enhance the interactiveness and persistence of human-machine interactions (Wang et al., 2018). QG benefits Question Answering (QA) models through data augmentation (Duan et al., 2017) and joint learning (Sun et al., 2019). It also plays an important role in education (Heilman and Smith, 2010) and clinical (Weizenbaum et al., 1966) systems.

Traditional Question Generation (TQG) is defined as the reverse task of QA, i.e., a passage and an answer (often a certain span from the passage) are provided as inputs, and the output is a question grounded in the input passage targeting on the given answer. When there is a sequence of answers, we can perform Sequential Question Generation (SQG) to produce a series of interconnected questions. Table 1 shows an example comparing the two tasks. Intuitively, questions in SQG are much more concise and we can regard them with given answers as QA-style conversations. Since it is more natural for human beings to test knowledge or seek information through coherent questions (Reddy et al., 2019), SQG has wide applications, e.g., enabling virtual assistants to ask questions based on previous discussions to get better user experiences.

SQG is a challenging task in two aspects. First, information omissions between questions lead to complex context dependencies. Second, there are frequently occurred coreference between questions. Prior works regarded SQG as a dialog generation task (namely conversational QG) where questions are generated **autoregressively** (recurrently), i.e., a new question is produced based on previous outputs. Although many powerful dialog generation models can be adopted to address the challenges mentioned above, there are two major obstacles. First, these models suffer from problems caused by error cascades. Empirical results from experiments reveal that the later generated questions tend to become shorter with lower quality, especially becoming more irrelevant to given answers, e.g., "Why?", "What else?". Second, models recurrently generating each question struggle to capture complex context dependencies, e.g., long-distance coreference. Essentially, SQG is rather different from dialog generation since all answers are given in advance and they act as strict semantic constraints during text generation.

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 225–237
July 5 - 10, 2020.

(1) A small boy named [John]$_1$ was at the park one day.
(2) He was [swinging]$_2$ [on the swings]$_3$ and [his friend]$_4$ named [Tim]$_5$ [played on the slide]$_6$.
(3) John wanted to play on the slide now.
(4) He asked Tim [if he could play on the slide]$_7$.
(5) Tim said [no]$_8$, and he cried.

Turn	TQG	SQG	Answer
1	Who was at the park?	Who was at the park?	John
2	What was John doing at the park?	What was he doing there?	swinging
3	Where was John swinging?	On what?	on the wings
4	Who was with John at the park?	Who was he with?	his friend
5	What is the name of John's friend?	Named?	Tim
6	What was Tim doing?	What was he doing?	played on the side
7	What did John asked Tim?	What did John asked him?	if he could play on the slide
8	What did Tim say to John?	What did he say?	no

Table 1: Comparison of Traditional Question Generation (TQG) and Sequential Question Generation (SQG). The given passage contains five sentences, and we mark the given answers in the passage as blue.

To deal with these problems, we perform SQG in a **semi-autoregressive** way. More specifically, we divide target questions into different groups (questions in the same group are closely-related) and generate all groups in parallel. Especially, our scenario becomes **non-autoregressive** if each group only contains a single question. Since we eliminate the recurrent dependencies between questions in different groups, the generation process is much faster and our model can better deal with the problems caused by error cascades. To get information for the generation process, we perform dual-graph interaction where a *passage-info graph* and an *answer-info graph* are constructed and iteratively updated with each other. The passage-info graph is used for better capturing context dependencies, and the answer-info graph is used to make generated questions more relevant to given answers with the help of our answer-aware attention mechanism. Besides, a coarse-to-fine text generation scenario is adopted for the coreference resolution between questions.

Prior works performed SQG on CoQA (Reddy et al., 2019), a high-quality dataset for conversational QA. As will be further illustrated, a number of data in CoQA are not suitable for SQG. Some researchers (Gao et al., 2019) directly discarded these data, but the remaining questions may become incoherent, e.g., the antecedent words for many pronouns are unclear. To this end, we build a new dataset from CoQA containing 81.9K relabeled questions. Above all, the main contributions of our work are:

- We build a new dataset containing 7.2K passages and 81.9K questions from CoQA. It is **the first** dataset specially built for SQG as far as we know.
- We perform semi-autoregressive SQG under dual-graph interaction. This is **the first time** that SQG is not regarded as a dialog generation task. We also propose an answer-aware attention mechanism and a coarse-to-fine generation scenario for better performance.
- We use extensive experiments to show that our model outperforms previous work by a substantial margin. Further analysis illustrated the impact of different components.

Dataset for this paper is available at `https://github.com/ChaiZ-pku/Sequential-QG`.

2 Related Work

2.1 Traditional Question Generation

TQG was traditionally tackled by rule-based methods (Lindberg et al., 2013; Mazidi and Nielsen, 2014; Hussein et al., 2014; Labutov et al., 2015), e.g., filling handcrafted templates under certain transformation rules. With the rise of data-driven learning approaches, neural networks (NN) have gradually taken the mainstream. Du et al. (2017) pioneered NN-based QG by adopting the Seq2seq architecture (Sutskever et al., 2014). Many ideas were proposed since then to make it more powerful, including answer position features (Zhou et al., 2017), specialized pointer mechanism (Zhao et al., 2018), self-attention (Scialom et al., 2019), answer separation (Kim et al., 2019), etc. In addition, enhancing the Seq2seq model into more complicated structures using variational inference, adversarial training and reinforcement learning (Yao et al., 2018; Kumar et al., 2019) have also gained much attention. There are also some works performing TQG under certain constraints, e.g., controlling the

topic (Hu et al., 2018) and difficulty (Gao et al., 2018) of questions. Besides, combining QG with QA (Wang et al., 2017; Tang et al., 2017; Sun et al., 2019) is also focused by many researchers.

2.2 Sequential Question Generation

As human beings tend to use coherent questions for knowledge testing or information seeking, SQG plays an important role in many applications. Prior works regarded SQG as a dialog generation task (namely conversational QA). Pan et al. (2019) pre-trained a model performing dialog generation, and then fine-tuned its parameters by reinforcement learning to make generated questions relevant to given answers. Gao et al. (2019) iteratively generated questions from previous outputs and leveraged off-the-shelf coreference resolution models to introduce a coreference loss. Besides, additional human annotations were performed on sentences from input passages for conversation flow modeling.

Since SQG is essentially different from dialog generation, we discard its dialog view and propose **the first** semi-autoregressive SQG model. Compared with using the additional human annotation in Gao et al. (2019), our dual-graph interaction deals with context dependencies automatically. Besides, our answer-aware attention mechanism is much simpler than the fine-tuning process in Pan et al. (2019) to make outputs more answer-relevant.

3 Dataset

As the reverse task of QA, QG is often performed on existing QA datasets, e.g., SQuAD (Rajpurkar et al., 2016), NewsQA (Trischler et al., 2016), etc. However, questions are independent in most QA datasets, making TQG the only choice. In recent years, the appearance of large-scale conversational QA datasets like CoQA (Reddy et al., 2019) and QuAC (Choi et al., 2018) makes it possible to train data-driven SQG models, and the CoQA dataset was widely adopted by prior works. Since the test set of CoQA is not released to the public, its training set (7.2K passages with 108.6K questions) was split into new training and validation set, and its validation set (0.5K passages with 8.0K questions) was used as the new test set.

Different from traditional QA datasets where the answers are certain spans from given passages, answers in CoQA are free-form text[1] with corresponding evidence highlighted in the passage. This brings a big trouble for QG. As an example, consider the yes/no questions counting for 19.8% among all questions. Given the answer "*yes*" and a corresponding evidence "*...the group first met on July 5 , 1967 on the campus of the Ohio state university...*", there are many potential outputs, e.g., "*Did the group first met in July?*", "*Was the group first met in Ohio state?*". When considering the context formed by previous questions, the potential outputs become even more (the original question in CoQA is "*Was it founded the same year?*"). When there are too many potential outputs with significantly different semantic meanings, training a converged QG model becomes extremely difficult. For this reason, Gao et al. (2019) directly discarded questions that cannot be answered by spans from passages. However, the remaining questions can become incoherent, e.g., antecedent words for many pronouns become unclear.

To this end, we build a new dataset from CoQA by preserving all 7.7K passages and rewriting all questions and answers. More specifically, we first discarded questions that are unsuitable for SQG. To do so, three annotators were hired to vote for the preservation/deletion of each question. A question is preserved if and only if it can be answered by a certain span from the input passage[2]. As a result, most deleted questions were yes/no questions and unanswerable questions. Besides, the kappa score between results given by different annotators was 0.83, indicating that there was a strong inter-agreement between annotators. For the remaining QA-pairs, we preserved their original order and replaced all answers by spans from input passages. After that, we rewrote all questions to make them coherent. To avoid over-editing, annotators were asked to modify as little as possible. It turned out that in most cases, they only needed to deal with coreference since the prototype of pronouns were no longer existed. To further guarantee the annotation quality, we hired another project manager who daily examined 10% of the annotations from each annotator and provided feedbacks. The annotation was considered valid only when the accuracy of examined results surpasses 95%. Our annotation process took 2 months, and we finally got a dataset containing 7.7K passage with 81.9K QA-pairs.

[1] Only 66.8% of the answers overlap with the passage after ignoring punctuations and case mismatches.

[2] Using certain spans from input passages (instead of free-formed text) as answers is a conversion in QG. In this way, the number of potential output questions is greatly reduced.

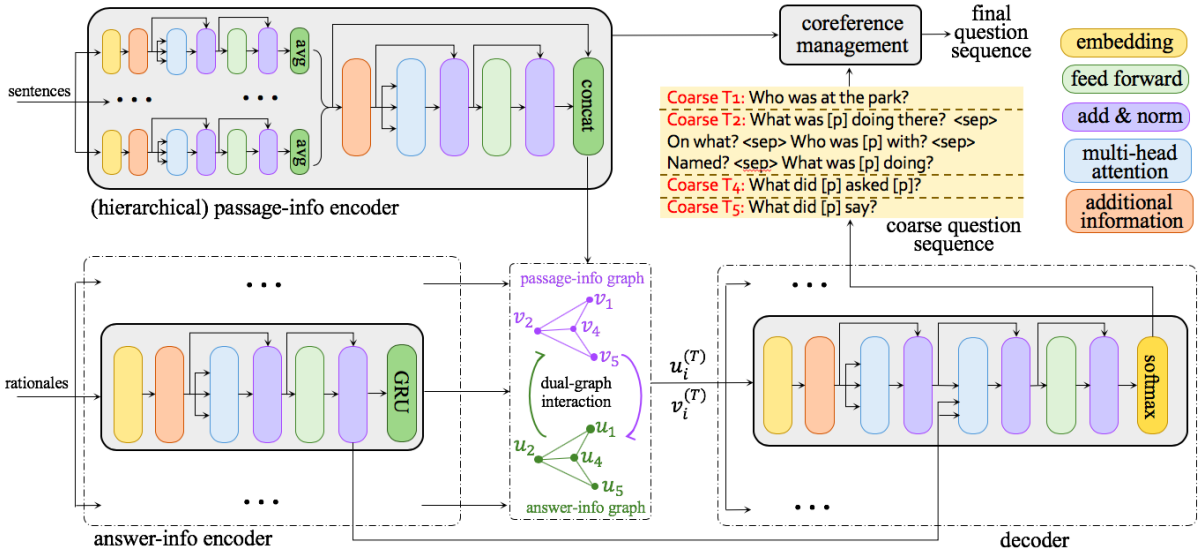

Figure 1: Architecture of our model. The example is corresponding with Table 1

4 Model

In this section, we formalize the SQG task and introduce our model in details. As shown in Figure 1, the model first builds a passage-info graph and an answer-info graph by its passage-info encoder and answer-info encoder respectively. After that, it performs dual-graph interaction to get representations for the decoder. Finally, different groups of questions are generated in parallel under a coarse-to-fine scenario. Both encoders and decoder take the form of Transformer architecture (Vaswani et al., 2017).

4.1 Problem Formalization

In SQG, we input a passage composed by n sentences $P = \{S_i\}_{i=1}^{n}$ and a sequence of l answers $\{A_i\}_{i=1}^{l}$, each A_i is a certain span of P. The target output is a series of questions $\{Q_i\}_{i=1}^{l}$, where Q_i can be answered by A_i according to the input passage P and previous QA-pairs.

As mentioned above, we perform SQG in an semi-autoregressive way, i.e., target questions are divided into into different groups. Ideally, questions in the same group are expected to be closely-related, while questions in different groups should be as independent as possible. Our model takes a simple but effective unsupervised question clustering method. The intuition is: if two answers come from the same sentence, the two corresponding questions are likely to be closely-related. More specifically, if the k-th sentence S_k contains p answers from $\{A_i\}_{i=1}^{l}$, we cluster them into an *answer-group* $\mathbb{G}_k^{ans} = \{A_{j_1}, A_{j_2}, ..., A_{j_p}\}$ where $j_1 < j_2 < ... < j_p$ are continuous indexes from $\{1, 2, ..., l\}$. By replacing each answer in $\mathbb{G}_k^{ans}$ with its corresponding question, we get a *question-group* $\mathbb{G}_k^{ques} = \{Q_{j_1}, Q_{j_2}, ..., Q_{j_p}\}$, and we further define a corresponding *target-output* T_k as "Q_{j_1} $[sep]$ Q_{j_2} $[sep]$ ... $[sep]$ Q_{j_p}" where "$[sep]$" is a special token. In Table 1, there are four target outputs T_1, T_2, T_4, T_5 (no T_3 since the third sentence in Table 1 do not contain any answer), T_2 is "*What was he doing there? [sep] On What? [sep] ... [sep] What was Tim doing?*" corresponding with the second sentence, and T_5 is "*What did he say?*" corresponding with the last sentence. Supposing there are m answer- and question-groups, then our model generates all the m target-outputs in parallel, i.e., all questions are generated in a semi-autoregressive way.

4.2 Passage-Info encoder

As shown in Figure 1, our passage-info encoder maps input sentences $\{S_i\}_{i=1}^{n}$ into their sentence representations $\{\boldsymbol{s}_i\}_{i=1}^{n}$ where every $\boldsymbol{s}_i \in \mathbb{R}^{2d_s}$. We regard each sentence as a sequence of words and replace each word by its pre-trained word embeddings (Mikolov et al., 2013) which is a dense vector. After that, the sequence of word embeddings is sent to a Transformer-encoder that outputs a corresponding sequence of vectors. By averaging these vectors, we get the *local representation* $\boldsymbol{s}_i^{local} \in \mathbb{R}^{d_s}$ of S_i.

After we get the local representations of all sentences $\{S_i\}_{i=1}^{n}$ in passage P, another Transformer-encoder is adopted to map the sequence $\{\boldsymbol{s}_i^{local}\}_{i=1}^{n}$ into $\{\boldsymbol{s}_i^{global}\}_{i=1}^{n}$, where $\boldsymbol{s}_i^{global} \in \mathbb{R}^{d_s}$ is called the

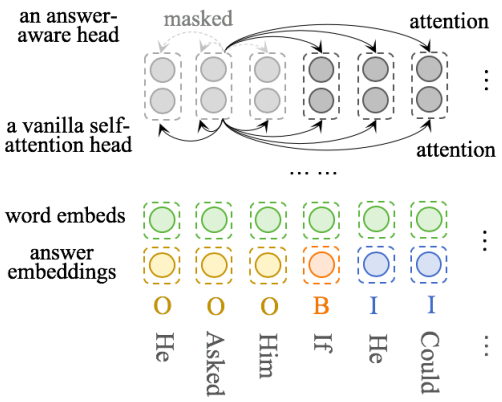

Figure 2: Illustration of answer embeddings and an answer-attention head for the forth sentence in Table 1.

global representation for S_i. In other words, the passage-info encoder takes a hiarachical structure. We expect the local and global representations capture intra- and inter- sentence context dependencies respectively, and the final representation for S_i is $\boldsymbol{s}_i = [\boldsymbol{s}_i^{local}; \boldsymbol{s}_i^{global}] \in \mathbb{R}^{2d_s}$.

4.3 Answer-Info Encoder

As described in Section 4.1, the input answers are split into m answer-groups. For $\mathbb{G}_k^{ans}$ corresponding with the k-th sentence of the input passage, we define $\{\mathbb{G}_k^{ans}, S_k\}$ as a "rationale" R_k, and further obtain its representation $\boldsymbol{r}_k \in \mathbb{R}^{2d_r}$ by our answer-info encoder, which is based on a Transformer-encoder regarding sentence S_k as its input.

To further consider information from $\mathbb{G}_k^{ans}$, two more components are added into the answer-info encoder, as shown in Figure 2. First, we adopt the *answer-tag features*. For each word w_i in sentence S_k, the embedding layer computes $[\boldsymbol{x}_i^w; \boldsymbol{x}_i^a] \in \mathbb{R}^{d_r}$ as its final embedding, where $\boldsymbol{x}_i^w$ is the pre-trained word embedding and $\boldsymbol{x}_i^a$ contains answer-tag features. More specifically, we give w_i a label from {O, B, I} if it is "outside", "the beginning of", "inside of" any answer from $\mathbb{G}_k^{ans}$, and use a vector corresponding with this label as $\boldsymbol{x}_i^a$. Second, we design the *answer-aware attention mechanism*. In the multi-head attention layer, there are not only l_h vanilla "self-attention heads", but also l_a "answer-aware heads" for each answer in $\mathbb{G}_k^{ans}$. In an answer-aware head corresponding with answer A, words not belonging to A are masked out during the attention mechanism. The output of the Transformer-encoder is a sequence of vectors $\boldsymbol{H}_k^{enc} = \{\boldsymbol{h}_k^{enc}\}$ ($\boldsymbol{h}_k^{enc} \in \mathbb{R}^{d_r}$) corresponding with the input word sequence from S_k.

After getting $\boldsymbol{H}_k^{enc}$, we further send the sequence of vectors to a bi-directional GRU network (Chung et al., 2014) and take its last hidden state as the final rationale embedding $\boldsymbol{r}_k \in \mathbb{R}^{2d_r}$.

4.4 Graph Construction

In our SQG task, the input passage contain n sentences, which can be represented by $\{\boldsymbol{s}_i\}_{i=1}^n \in \mathbb{R}^{2d_s}$ leveraging the passage-info encoder. Among all input sentences, only m of them contain certain answers ($m \leq n$), and we further define m rationales based on these sentences, $\{\mathbb{G}_{F(j)}^{ans}, S_{F(j)}\}_{j=1}^m$, where the j-th rationale ($j \in \{1, 2, ..., m\}$) corresponds with the $F(j)$-th sentence of the input passage ($F(j) \in \{1, 2, ..., n\}$). For the example in Table 1, $n = 5, m = 4, F(j)$ maps $\{1, 2, 3, 4\}$ into $\{1, 2, 4, 5\}$ respectively. Using the answer-info encoder, we can get representations $\{\boldsymbol{r}_{F(j)}\}_{j=1}^m \in \mathbb{R}^{2d_s}$ for all rationales.

We further build a passage-info graph $\mathcal{V}$ and an answer-info graph $\mathcal{U}$ based on these representations. For the rationale corresponding with the k-th sentence of the input passage, we add node u_k, v_k in graph $\mathcal{U}, \mathcal{V}$ respectively. For the example in Table 1, $\mathcal{U}$ is compused by $\{u_1, u_2, u_4, u_5\}$ and $\mathcal{V}$ is compused by $\{v_1, v_2, v_4, v_5\}$, as shown in Figure 1. The initial representation for u_k is computed by:

$$\boldsymbol{u}_k^{(0)} = ReLU(\boldsymbol{W}_u[\boldsymbol{r}_k; \boldsymbol{e}_k] + \boldsymbol{b}_u) \in \mathbb{R}^{d_g} \quad (1)$$

where $\boldsymbol{r}_k \in \mathbb{R}^{2d_r}$ is the rationale representation, $\boldsymbol{e}_k \in \mathbb{R}^{d_e}$ is the embedding of index k, and $\boldsymbol{W}_u \in \mathbb{R}^{(d_e+2d_r)\times d_g}, \boldsymbol{b}_u \in \mathbb{R}^{d_g}$ are trainable parameters. And the initial representation for v_k is:

$$\boldsymbol{v}_k^{(0)} = ReLU(\boldsymbol{W}_v[\boldsymbol{s}_k; \boldsymbol{e}_k] + \boldsymbol{b}_v) \in \mathbb{R}^{d_g} \quad (2)$$

where $\boldsymbol{s}_k \in \mathbb{R}^{2d_s}$ is the sentence representation and $\boldsymbol{W}_v \in \mathbb{R}^{(d_e+2d_s)\times d_g}, \boldsymbol{b}_v \in \mathbb{R}^{d_g}$ are parameters.

After adding these points, there are m nodes in $\mathcal{U}$ and $\mathcal{V}$ respectively. For $u_i, u_j \in \mathcal{U}$ corresponding with the i-th, j-th input sentences respectively, we add an edge between them if $|i - j| < \delta$ (δ is a hyper-parameter). Similarly, we add edges into $\mathcal{V}$ and the two graphs are isomorphic.

4.5 Dual-Graph Interaction

In our answer-info graph $\mathcal{U}$, node representations contain information focused on input answers. In the passage-info graph $\mathcal{V}$, node representations capture inter- and intra-sentence context dependencies. As mentioned above, a good question should be

answer-relevant as well as capturing complex context dependencies. So we should combine information in both $\mathcal{U}$ and $\mathcal{V}$. Our dual-graph interaction is a process where $\mathcal{U}$ and $\mathcal{V}$ iteratively update node representations with each other. At time step t, representations $\boldsymbol{u}_i^{(t-1)}, \boldsymbol{v}_i^{(t-1)}$ are updated into $\boldsymbol{u}_i^{(t)}, \boldsymbol{v}_i^{(t)}$ respectively under three steps.

First, we introduce the *information transfer step*. Taking $\mathcal{U}$ as an example. Each $\boldsymbol{u}_i^{(t-1)}$ receives $\boldsymbol{a}_i^{(t)}$ from its neighbors (two nodes are neighbors if there is an edge between them) by:

$$\boldsymbol{a}_i^{(t)} = \sum_{u_j \in \mathcal{N}(u_i)} \boldsymbol{W}_{ij}\, \boldsymbol{u}_j^{(t-1)} + \boldsymbol{b}_{ij} \tag{3}$$

where $\mathcal{N}(u_i)$ is composed by all neighbors of node u_i and $\boldsymbol{W}_{ij} \in \mathbb{R}^{d_g \times d_g}$, $\boldsymbol{b}_{ij} \in \mathbb{R}^{d_g}$ are parameters controlling the information transfer. For u_i, u_j and $u_{i'}, u_{j'}$ whose $|i-j| = |i'-j'|$, we use the same $\boldsymbol{W}$ and $\boldsymbol{b}$. In other words, we can first create a sequence of matrices $\{\boldsymbol{W}_1, \boldsymbol{W}_2, ...\} \in \mathbb{R}^{d_g \times d_g}$ and vectors $\{\boldsymbol{b}_1, \boldsymbol{b}_2, ...\} \in \mathbb{R}^{d_g}$, and then use $|i-j|$ as the index to retrieve the corresponding $\boldsymbol{W}_{ij}, \boldsymbol{b}_{ij}$. For graph $\mathcal{V}$, we similarly compute

$$\tilde{\boldsymbol{a}}_i^{(t)} = \sum_{v_j \in \mathcal{N}(v_i)} \tilde{\boldsymbol{W}}_{ij}\, \boldsymbol{v}_j^{(t-1)} + \tilde{\boldsymbol{b}}_{ij} \tag{4}$$

In the second step, we *compute multiple gates*. For each $\boldsymbol{u}_i^{(t-1)}$ in $\mathcal{U}$, we compute an "update gate" $\boldsymbol{y}_i^{(t)}$ and a "reset gate" $\boldsymbol{z}_i^{(t)}$ by:

$$\begin{aligned} \boldsymbol{y}_i^{(t)} &= \sigma(\boldsymbol{W}_y[\boldsymbol{a}_i^{(t)}; \boldsymbol{u}_i^{(t-1)}]) \\ \boldsymbol{z}_i^{(t)} &= \sigma(\boldsymbol{W}_z[\boldsymbol{a}_i^{(t)}; \boldsymbol{u}_i^{(t-1)}]) \end{aligned} \tag{5}$$

where $\boldsymbol{W}_y, \boldsymbol{W}_z \in \mathbb{R}^{2d_g \times d_g}$ are paramenters. Similarly, for each $\boldsymbol{v}_i^{(t-1)}$ in $\mathcal{V}$ we compute:

$$\begin{aligned} \tilde{\boldsymbol{y}}_i^{(t)} &= \sigma(\tilde{\boldsymbol{W}}_y[\tilde{\boldsymbol{a}}_i^{(t)}; \boldsymbol{v}_i^{(t-1)}]) \\ \tilde{\boldsymbol{z}}_i^{(t)} &= \sigma(\tilde{\boldsymbol{W}}_z[\tilde{\boldsymbol{a}}_i^{(t)}; \boldsymbol{v}_i^{(t-1)}]) \end{aligned} \tag{6}$$

Finally, we perform the *information interaction*, where each graph updates its node representations under the control of gates computed by **the other graph**. More specifically, node representations are updated by:

$$\begin{aligned} \boldsymbol{u}_i^{(t)} =& \tilde{\boldsymbol{z}}_i^{(t)} \odot \boldsymbol{u}_i^{(t-1)} + (\boldsymbol{1} - \tilde{\boldsymbol{z}}_i^{(t)}) \odot \\ & tanh(\boldsymbol{W}_a[\boldsymbol{a}_i^{(t)}; \tilde{\boldsymbol{y}}_i^{(t)} \odot \boldsymbol{u}_i^{(t-1)}]) \\ \boldsymbol{v}_i^{(t)} =& \boldsymbol{z}_i^{(t)} \odot \boldsymbol{v}_i^{(t-1)} + (\boldsymbol{1} - \boldsymbol{z}_i^{(t)}) \odot \\ & tanh(\tilde{\boldsymbol{W}}_a[\tilde{\boldsymbol{a}}_i^{(t)}; \boldsymbol{y}_i^{(t)} \odot \boldsymbol{v}_i^{(t-1)}]) \end{aligned} \tag{7}$$

The idea of using gates computed by the other graph to update node representations in each graph enables the information in input passage and answers interact more frequently, both of which act as strong constraints to the output questions.

By iteratively performing the three steps for T times, we get the final representations $\boldsymbol{u}_i^{(T)}$ and $\boldsymbol{v}_i^{(T)}$ for $u_i \in \mathcal{U}$ and $v_i \in \mathcal{V}$.

4.6 Decoder

For the k-th input sentence S_k containing certain answers, our decoder generates the corresponding target-output T_k. As mentioned above, the generation process of all target-outputs are independent. The decoder is based on the Transformer-decoder containing a (masked) multi-head self-attention layer, a multi-head encoder-attention layer, a feed-forward projection layer and the softmax layer. To compute keys and values for the multi-head encoder-attention layer, it leverages the outputs from our answer-info encoder, i.e., it uses $\boldsymbol{H}_k^{enc}$ described in Section 4.3 to generate T_k corresponding with the k-th sentence.

To generate coherent questions, we need to capture the context dependencies between input answers and passages. To this end, both $\boldsymbol{u}_k^{(T)}$ and $\boldsymbol{v}_k^{(T)}$, which comes from the dual-graph interaction process, are used as additional inputs for generating T_k. First, they are concatenated with the output of each head from both (masked) multi-head self-attention layer and multi-head encoder-attention layer before sending to the next layer. Second, they are concatenated with inputs of the feed-forward projection layer. The two representations are also expected to make generated questions more relevant to given inputs.

4.7 Coarse-To-Fine Generation

Since the semi-autoregressive generation scenario makes it more challenging to deal with coreferences between questions (especially questions in different groups), we perform question generation in a coarse-to-fine manner. The decoder only needs to generate "coarse questions" where all pronouns are replaced by a placeholder "*[p]*". To get final results, we use an additional pre-trained coreference resolution model to fill pronouns into different placeholders. To make a fair comparison, we use the coreference resolution model (Clark and Manning, 2016) adopted by prior works *CoreNQG* (Du and Cardie, 2018) and *CorefNet* (Gao et al., 2019).

Model	$BLEU_1$	$BLEU_2$	$BLEU_3$	ROUGE	METEOR	Length
Seq2seq (Du et al., 2017)	28.72	10.16	6.30	31.75	13.10	5.78
CopyNet (See et al., 2017)	29.40	12.14	6.53	33.71	14.20	5.77
CoreNQG (Du and Cardie, 2018)	33.84	14.69	8.72	34.38	14.05	**6.08**
VHRED (Serban et al., 2017)	30.51	11.95	6.94	31.93	12.42	4.83
HRAN (Xing et al., 2018)	30.18	12.53	7.65	35.06	12.95	5.02
ReDR (Pan et al., 2019)	30.84	15.17	9.81	35.58	15.41	5.58
CorefNet (Gao et al., 2019)	32.72	16.01	10.97	37.48	16.09	5.96
Ours	**35.70**	**19.64**	**12.06**	**38.15**	**17.26**	6.03

Table 2: Experimental results. In each column, we bold / underline the best performance over all / baseline methods, respectively. Under the evaluation of BLEU, ROUGE-L and METEOR, our model differs from others (except the METEOR score of CorefNet) significantly based on the one-side paired t-test with $p < 0.05$.

5 Experiments

In this section, we first introduce the three kinds of baselines. After that, we compare and analyse the results of different models under both automatic and human evaluation metrics.

5.1 Baselines

We compared our model with seven baselines that can be divided into three groups. First, we used three TQG models: the *Seq2seq (Du et al., 2017)* model which pioneered NN-based QG, the *CopyNet (See et al., 2017)* model that introduced pointer mechanism, and *CoreNQG (Du and Cardie, 2018)* which used hybrid features (word, answer and coreference embeddings) for encoder and adopted copy mechanism for decoder. Second, since prior works regarded SQG as a conversation generation task, we directly used two powerful multi-turn dialog systems: the latent variable hierarchical recurrent encoder-decoder architecture *VHRED (Serban et al., 2017)*, and the hierarchical recurrent attention architecture *HRAN (Xing et al., 2018)*. Third, we used prior works mentioned above. For Pan et al. (2019), we adopted the *ReDR* model which had the best performance. For Gao et al. (2019), we used the *CorefNet* model. Although a *CFNet* in this paper got better results, it required additional human annotations denoting the relationship between input sentences and target questions. So it is unfair to compare *CFNet* with other methods.

It is worth mentioning that when generating questions using the second and third groups of baselines, **only previously generated** outputs were used as dialog history, i.e., the gold standard questions are **remain unknown** (in some prior works, they were directly used as dialog history, which we think is inappropriate in practice).

	SQuAD	CoQA	Ours
Passage	117	271	271
Question	10.1	5.5	6.6
Answer	3.2	2.7	3.2

Table 3: Average number of words in passage, question and answer in different datasets.

5.2 Automatic Evaluation Metrics

Following the conventions, we used BLEU (Papineni et al., 2002), ROUGE-L (Lin, 2004) and METEOR (Lavie and Agarwal, 2007) as automatic evaluation metrics. We also computed the average word-number of generated questions. As shown in Table 2, our semi-autoregressive model outperformed other methods substantially.

When we focus on the second and third groups of baselines regarding SQG as multi-turn dialog generation tasks, we can find that models from the third group are more powerful since they make better use of information from input passages. Besides, models from the second group tend to generate shortest questions. Finally, similar to the problem that dialog systems often generate dull and responses, these models also suffer from producing general but meaningless questions like "What?", "How?", "And else?".

When we compare the first and third groups of baselines (which are all QG models), it is not surprising that SQG models show more advantages than TQG models, as they take the relationships between questions into consideration. Besides, CorefNet gets better performance among all baselines, especially ReDR. This indicates that comparing with implicitly performing reinforcement learning through QA models, explicitly using target answers as inputs can be more effective.

	CoreNQG	CorefNet	Ours
Fluency	2.36	**2.51**	2.44
Coherence	1.53	2.04	**2.17**
Coreference	1.15	**1.56**	1.54
Answerability	1.12	1.18	**1.45**
Relevance	1.47	1.24	**1.62**

Table 4: Human evaluation results. Scores of each metric ranges between 1 to 3 and larger scores are better.

	$BLEU_3$	ROUGE	METEOR
No interact	11.35	37.31	17.05
Uni-graph	9.86	36.44	15.87
Uni-heads	10.33	37.48	16.24
No co2fine	11.75	37.92	17.17
Non-auto	7.79	33.62	14.83
Ours	12.06	38.15	17.26

Table 5: Results for ablation tests.

Note that if we directly compare the performance between SQG task and TQG task under the same model (e.g., the Seq2seq model), evaluation scores for TQG tasks are much higher, which is not surprising since SQG is harder than TQG dealing with dependencies between questions. Another fact lies in the computation of automatic evaluation metrics. As shown in Table 2, questions in SQG datasets are much shorter than TQG. Since our automatic evaluation metrics are based on n-gram overlaps between generated and gold standard questions, the scores significantly go down with the growth of n (for this reason, the $BLEU_4$ scores are not listed in Table 2). This also illustrates the importance of performing human evaluation.

5.3 Human Evaluation

It is generally acknowledged that automatic evaluation metrics are far from enough for SQG. So we perform human evaluation in five aspects. *Fluency* measures if a question is grammatically correct and is fluent to read. *Coherence* measures if a question is coherent with previous ones. *Coreference* measures if a question uses correct pronouns. *Answerability* measures if a question is targeting on the given answer. *Relevance* measures if a question is grounded in the given passage. Since performing human evaluation is rather expensive and time-consuming, we picked up the best TQG model (CoreNQG), SQG model (CorefNet) to compare with our model. We randomly selected 20 passages from the test set with 207 given answers and asked 10 native speakers to evaluate the outputs of each model independently. Under each aspect, reviewers are asked to choose a score from $\{1, 2, 3\}$, where 3 indicates the best quality.

The average scores for each evaluation metric are shown in Table 4. We can find that our model gets the best or competitive performance in each metric. When it comes to fluency, all models get high performance, and the CorefNet that outputs shortest questions gets the best score. As for coherence, CoreNQG gets poor results since it generates questions independently. When it comes to coreference, our model only slightly lower than CorefNet, which added direct supervision to attention weights by a coreference resolution model. Finally, our model gets the best performance on both answerabity and relevance. However, it is worth noticing that all models get rather poor performances under these two aspects, indicating that making a concise question meaningful (i.e., targeting on given answers) with more information from input passage (i.e., performing proper information elimination) is a major challenge in SQG. Besides, as pointed out by Table 3, questions in our SQG dataset are significantly shorter compared with TQG dataset, making subtle errors much easier to be noticed.

6 Analysis

6.1 Ablation Test

In this section, we perform ablation test to verify the influence of different components in our model. First, we modify Equation 7 into

$$\begin{aligned} \boldsymbol{u}_i^{(t)} &= \boldsymbol{z}_i^{(t)} \odot \boldsymbol{u}_i^{(t-1)} + (\mathbf{1} - \boldsymbol{z}_i^{(t)}) \odot \\ &\quad tanh(\boldsymbol{W}_a[\boldsymbol{a}_i^{(t)}; \boldsymbol{y}_i^{(t)} \odot \boldsymbol{u}_i^{(t-1)}]) \\ \boldsymbol{v}_i^{(t)} &= \tilde{\boldsymbol{z}}_i^{(t)} \odot \boldsymbol{v}_i^{(t-1)} + (\mathbf{1} - \tilde{\boldsymbol{z}}_i{}^{(t)}) \odot \\ &\quad tanh(\tilde{\boldsymbol{W}}_a[\tilde{\boldsymbol{a}}_i^{(t)}; \tilde{\boldsymbol{y}}_i^{(t)} \odot \boldsymbol{v}_i^{(t-1)}]) \end{aligned} \tag{8}$$

to get the *no interact* model, i.e., two graphs are independently updated without any interaction. Second, we build a *uni-graph* model by removing the passage-info encoder (the remaining rationale graph is updated similarly to Li et al. (2015)). Third, we discard the attention-aware heads in the rationale encoder to get a *uni-heads* model. Then, we build the *no co2fine* model without the coarse-to-fine generation scenario. Finally, we build a *non-auto* model that performs SQG in an non-autoregressive way, i.e., each question is generated in parallel.

Peter was a very sad puppy. He had been inside of the pet store for a very long time. In fact, he had been there for [three months]$_1$! Peter had seen many other puppies find a person; he began to wonder why he could not get one. He thought that [maybe his fur was not pretty enough or maybe his bark was not loud enough]$_2$. He tried and tried to please every person who came to the store, but they all picked smaller puppies. However, one day all of this changed. [Sammie]$_3$ came into the store looking for [a golden puppy]$_4$. She wanted a puppy she could snuggle with. It so happened that Peter was very sad and tired that day. Sammie came to hold him. Peter wanted to show off [his bark]$_5$, but he was [too tired]$_6$. He [fell right to sleep]$_7$. Sammie loved him at once and loved holding him in her arms. Sammie took [Peter]$_8$ home that day, and they made lots of fun memories.

Turn	Gold Standard	CorefNet	Ours
1	How long was Peter at pet store?	How long he had been there?	How long was Peter there?
2	Why couldn't he get someone?	What his fur was?	What did he thought?
3	Who came into the store?	Who came into the store?	Who came into the store?
4	What for?	What was Sammie looking?	Who was she looking for?
5	What did peter wanted to show off?	What Peter wanted show off?	What he show off?
6	Why not?	Why he wanted?	What was he?
7	What did he do with her?	And else?	What did he do?
8	Who did she take?	Who was Sammie took?	What Sammie took that day?

Table 6: Example outputs from different models. We mark the given answers in the passage as blue.

As shown in Table 5, each component in our model plays an important part. Results for the *no interact* model indicate that compared with independently updating the passage-info graph and answer-info graph, making these information more interacted by our dual-graph interaction scenario is more powerful. Not surprisingly, the *uni-graph* model removing the passage encoder (i.e., less focusing on context dependencies between sentences from input passage), and the *uni-heads* model discarding our answer-aware attention mechanism (i.e., less focusing on given answers) get significant worse performance compared with our full model. Besides, our coarse-to-fine scenario helps to better deal with the dependencies between questions since there are widespread coreferences. Finally, although the architecture of *non-auto* model is a special case of our model where each group only contains a single question, the performance drops significantly, indicating the importance of using semi-autoregressive generation. However, the dual-graph interaction still makes its performance better than the Seq2seq and CopyNet in Table 2.

6.2 Running Examples

In Table 6, we present some generated examples comparing our model and the strongest baseline CorefNet. On the one hand, our model performs better than CorefNet, especially that the output questions are more targeting on given answers (turn 2, 6, 7). It also correctly deals with coreferences (e.g., distinguishing "Peter" and "Sammie"). On the other hand, the generated questions have poor quality when gold standard questions involve more reasoning (turn 2, 6). Besides, the gold standard questions are more concise as well (turn 4, 6).

7 Conclusion

In this paper, we focus on SQG which is an important yet challenging task. Different from prior works regarding SQG as a dialog generation task, we propose the first semi-autoregressive SQG model, which divides questions into different groups and further generates each group of closely-related questions in parallel. During this process, we first build a passage-info graph, an answer-info graph, and then perform dual-graph interaction to get representations capturing the context dependencies between passages and questions. These representations are further used during our coarse-to-fine generation process. To perform experiments, we analyze the limitation of existing datasets and create the first dataset specially used for SQG containing 81.9K questions. Experimental results show that our model outperforms previous works by a substantial margin.

For future works, the major challenge is generating more meaningful, informative but concise questions. Besides, more powerful question clustering and coarse-to-fine generation scenarios are also worth exploration. Finally, performing SQG on other types of inputs, e.g., images and knowledge graphs, is an interesting topic.

Acknowledgments

This work was supported by National Natural Science Foundation of China (61772036) and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). We thank the anonymous reviewers for their helpful comments. Xiaojun Wan is the corresponding author.

References

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wentau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. Quac: Question answering in context. *arXiv preprint arXiv:1808.07036*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Kevin Clark and Christopher D Manning. 2016. Deep reinforcement learning for mention-ranking coreference models. *arXiv preprint arXiv:1609.08667*.

Xinya Du and Claire Cardie. 2018. Harvesting paragraph-level question-answer pairs from wikipedia. *arXiv preprint arXiv:1805.05942*.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. *arXiv preprint arXiv:1705.00106*.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 866–874.

Yifan Gao, Piji Li, Irwin King, and Michael R Lyu. 2019. Interconnected question generation with coreference alignment and conversation flow modeling. *arXiv preprint arXiv:1906.06893*.

Yifan Gao, Jianan Wang, Lidong Bing, Irwin King, and Michael R Lyu. 2018. Difficulty controllable question generation for reading comprehension. *arXiv preprint arXiv:1807.03586*.

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. Association for Computational Linguistics.

Wenpeng Hu, Bing Liu, Jinwen Ma, Dongyan Zhao, and Rui Yan. 2018. Aspect-based question generation.

Hafedh Hussein, Mohammed Elmogy, and Shawkat Guirguis. 2014. Automatic english question generation system based on template driven scheme. *International Journal of Computer Science Issues (IJCSI)*, 11(6):45.

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.

Vishwajeet Kumar, Ganesh Ramakrishnan, and Yuan-Fang Li. 2019. Putting the horse before the cart: A generator-evaluator framework for question generation from text. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 812–821.

Igor Labutov, Sumit Basu, and Lucy Vanderwende. 2015. Deep questions without deep understanding. 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: Long Papers)*, volume 1, pages 889–898.

Alon Lavie and Abhaya Agarwal. 2007. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231. Association for Computational Linguistics.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2015. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

David Lindberg, Fred Popowich, John Nesbit, and Phil Winne. 2013. Generating natural language questions to support learning on-line. In *Proceedings of the 14th European Workshop on Natural Language Generation*, pages 105–114.

Karen Mazidi and Rodney D Nielsen. 2014. Linguistic considerations in automatic question generation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 321–326.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Margaret Mitchell, Xiaodong He, and Lucy Vanderwende. 2016. Generating natural questions about an image. *arXiv preprint arXiv:1603.06059*.

Boyuan Pan, Hao Li, Ziyu Yao, Deng Cai, and Huan Sun. 2019. Reinforced dynamic reasoning for conversational question generation. *arXiv preprint arXiv:1907.12667*.

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*, pages 311–318. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2019. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

Thomas Scialom, Benjamin Piwowarski, and Jacopo Staiano. 2019. Self-attention architectures for answer-agnostic neural question generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6027–6032.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. *arXiv preprint arXiv:1603.06807*.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Yibo Sun, Duyu Tang, Nan Duan, Shujie Liu, Zhao Yan, Ming Zhou, Yuanhua Lv, Wenpeng Yin, Xiaocheng Feng, Bing Qin, et al. 2019. Joint learning of question answering and question generation. *IEEE Transactions on Knowledge and Data Engineering*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Duyu Tang, Nan Duan, Tao Qin, Zhao Yan, and Ming Zhou. 2017. Question answering and question generation as dual tasks. *arXiv preprint arXiv:1706.02027*.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Tong Wang, Xingdi Yuan, and Adam Trischler. 2017. A joint model for question answering and question generation. *arXiv preprint arXiv:1706.01450*.

Yansen Wang, Chenyi Liu, Minlie Huang, and Liqiang Nie. 2018. Learning to ask questions in open-domain conversational systems with typed decoders.

Joseph Weizenbaum et al. 1966. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.

Chen Xing, Yu Wu, Wei Wu, Yalou Huang, and Ming Zhou. 2018. Hierarchical recurrent attention network for response generation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Kaichun Yao, Libo Zhang, Tiejian Luo, Lili Tao, and Yanjun Wu. 2018. Teaching machines to ask questions. In *IJCAI*, pages 4546–4552.

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.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017. Neural question generation from text: A preliminary study. In *National CCF Conference on Natural Language Processing and Chinese Computing*, pages 662–671. Springer.

A Examples of Data Labeling

In Table 7, we use a typical example to show how we relabeled CoQA. As introduced in our paper, we first deleted questions that cannot be answered by certain span from the passage. In Table 7, we deleted QA-pairs in turn 15, 18, 19 since they are yes/no questions, turn 3, 16 since the answer "female" is not a span from the input passage, and turn 13 since its answer is scattered in the sentence "Some of his cats have orange fur, some have black fur, some are spotted and one is white".

After deleting questions that are not suitable for SQG, we replaced the remaining answers into certain spans from the input passage. As shown in Table 7, in most cases the original answers were already a certain span. We slightly modified answers in turn 2, 7 from "Eight", "Three" into "8", "3" respectively. Finally, we rewrote all remaining questions to make them coherent. During this process, we mainly deal with information omission and coreference. In our example, we added a word "feline" into questions in turn 14 since the question 13 was deleted.

B Details of Experiments

We used the 200-dimentional pre-trained GloVe word embeddings [3] as initial value of word embeddings. During the training process, these embeddings were further fine-tuned. The NLTK[4] package was used for sentence splitting and word tokenization. In our model, we set d_s, d_r, d_g to 200, 256 and 128. For the passage-info encoder, we used 16 heads in the multil-attention layer. For the answer-info encoder, we used 8 vanilla self-attention heads and additional 6 answer-aware heads for each answer. To construct the two graphs, we set δ into 3. In our dual-graph interaction, we set T into 4.

To train our model, we used an Adam optimizer with momentums $\beta_1 = 0.9$, $\beta_2 = 0.99$ and $\epsilon = 10^{-8}$ to minimize the loss function. We varied the learning rate throughout training, including a warm-up step and a decreasing step similar to the original Transformer. Besides, we applied dropout between 0.4 and 0.5 to prevent over-fitting. Our model was trained on two Nvidia RTX 2080Ti graphics cards.

Since we noticed that the available baseline codes used different scripts to compute BLEU, ROUGE and METEOR, we used new scripts[5] to compute the evaluation metrics in this paper.

[3] https://nlp.stanford.edu/projects/glove/

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

[5] https://github.com/tylin/coco-caption/tree/master/pycocoevalcap

Brendan loves cats. He owns 8 cats. He has 7 girl cats and only 1 boy cat. Brendan brushes the cats' hair every day. He makes sure to feed them every morning and evening and always checks to see if the cats have water. Sometimes he feeds them special treats because he loves them. Each cat gets 3 treats. He doesn't give them food like chips and cake and candy, because those foods aren't good for cats. He likes to play with the cats. The cats like to chase balls of paper that Brendan makes for them. Some of his cats have orange fur, some have black fur, some are spotted and one is white. The white cat is Brendan's favorite. She is the first cat he owned. Her name is Snowball. When he first got Snowball she was a kitten. His other cats are named Fluffy, Salem, Jackie, Cola, Snickers, Pumpkin and Whiskers.		
turn	Original QA-Pairs	New QA-Pairs
1	What does he care for? (cats)	What does he care for? (cats)
2	How many does he have? (Eight)	How many does he have? (8)
3	Are there more males or females? (females)	Deleted
4	How many? (7 girl cats and only 1 boy cat)	How many males and females? (7 girl cats and only 1 boy cat)
5	What is groomed? (cat's hair)	What is groomed? (cat's hair)
6	What do they get fed? (treats)	What do they get fed? (treats)
7	How many? (Three)	How many? (3)
8	Why (because he loves them)	Why (because he loves them)
9	What foods are avoided? (chips and cake and candy)	What foods are avoided? (chips and cake and candy)
10	Why? (because those foods aren't good for cats)	Why? (because those foods aren't good for cats)
11	What toys do they like? (balls of paper)	What toys do they like? (balls of paper)
12	Who creates them? (Brendan)	Who creates them? (Brendan)
13	What colors are the felines? (orange, black, spotted, and white)	Deleted
14	Which is the most liked? (The white cat)	Which is the most liked? (The white cat)
15	Is this his original one? (yes)	Deleted
16	What is its gender? (female)	Deleted
17	What does he call it? (Snowball)	What does he call it? (Snowball)
18	Is there one called Binky? (No)	Deleted
19	How about Scruff? (No)	Deleted

Table 7: Example for data labeling.