Consecutive Question Generation via Dynamic Multitask Learning

Anonymous ACL submission

Abstract

In this paper, we propose the task of consecutive question generation (CQG), which generates a set of logically related question-answer pairs to understand a whole passage, with a comprehensive consideration of the aspects including accuracy, coverage, and informativeness. To achieve this, we first examine the four key elements of CQG, i.e., question, answer, rationale[1], and context history[2], and propose a novel dynamic multitask framework with one main task generating a question-answer pair, and four auxiliary tasks generating other elements. It directly helps the model generate good questions through both joint training and self-reranking. At the same time, to fully explore the worth-asking information in a given passage, we make use of the reranking losses to sample the rationales and search for the best question series globally. Finally, we measure our strategy by QA data augmentation and manual evaluation, as well as a novel application of generated question-answer pairs on DocNLI. We prove that our strategy can improve question generation significantly and benefit multiple related NLP tasks.

1 Introduction

Question Generation (QG) is an important and promising task in natural language generation (NLG). It has long served as an effective way to improve other NLP tasks. The applications of synthetic questions have expanded from QA data augmentation (Duan et al., 2017; Lewis et al., 2021) to building tutoring or dialogue systems (Lindberg et al., 2013; Bordes and Weston, 2017), self-assessing the ability of language models (Sun et al., 2019), and checking the faithfulness of an abstract summary (Durmus et al., 2020), etc.

Traditionally, syntax-based methods such as semantic parsing are commonly adopted to synthesize questions (Berant et al., 2013; Khullar et al., 2018). Recently, transformer-based pre-trained language models (Vaswani et al., 2017; Devlin et al., 2019) are widely used to generate questions. Most of these works are two-step QG methods (Sun et al., 2018; Rennie et al., 2020), which rely on ground-truth or pre-extracted answers (Wang et al., 2019; Jia et al., 2020) and generate questions independently (Puri et al., 2020; Bartolo et al., 2021). However, in real scenarios such as daily conversations or reading comprehension, we usually raise several questions consecutively to understand a whole story. Current QG methods are inadequate to generate such questions, as Table 1 shows. We can see that there are no logical connections between the questions (e.g., Q3 and Q1) and pre-extracted answers also lead to simplicity (e.g., Q1) and inconsistency (e.g., Q3).

Today is Jessica's 80th birthday. Her daughter Mela and Mela's husband Josh is coming over to the birthday party...	
*Q1: **Who is her daughter?***	*A1: **Mela.***
*Q2: **Who is Josh?***	*A2: **Mela's husband.***
*Q3: **Who has a birthday party?***	*A3: **Mela.***

Table 1: Example QG results using a two-step inconsecutive method based on extractive answers.

In such cases, we propose the task of consecutive question generation (CQG), which automatically produces a set of well-ordered and logically related question-answer (Q-A) pairs to help understand a given passage (or story). Table 2 shows several "ideal" questions which are mutually connected and cover diverse information in the text. To achieve this, unlike traditional QG methods, which mainly focus on "what are good questions", our CQG also requires a model to automatically find "which information in a text is worth-asking". Additionally, since we pose questions not only to get separate information, but to understand a

[1]The sentence based on which a question is generated.

[2]The coverage of all previous rationales, representing the background information of the current question series.

whole story, we propose three key qualities simultaneously to evaluate consecutive questions, i.e., accuracy, coverage, and informativeness.

With these demands, we propose an integrated dynamic multitask framework, with five unified Seq2Seq generation tasks. One main task generates Q-A pairs and four auxiliary tasks make full use of the generation of four key CQG elements (i.e., question, answer, rationale, and context history). We link the qualities of key aspects with the losses of four auxiliary tasks respectively. Based on it, we then design four distinct methods to improve the model performance from all aspects and from all stages during training and inference.

The five tasks are jointly trained in one model to help it learn from different views. In inference, the main task generates candidates and then the auxiliary tasks self-rerank them, improving Q-A accuracy, coverage, and informativeness all-roundly. To fully exploit the worth-asking information in each sentence and generate questions properly and dynamically, we propose a novel rationale sampling method and sentence-level beam-search. We recompose the context history reranking losses to measure the information in each rationale, and then design a sample probability to guarantee that the more information a rationale leaves, the more likely it is asked once again. To relieve the error cascade and guide the direction of a Q-A flow, we reinvent beam-search to sentence-level, which rearranges the total reranking results and seeks the global optimum Q-A series for a whole passage.

Finally, we conduct abundant experiments on QA augmentation and make a manual evaluation. We further propose a novel method for document-level NLI task (Yin et al., 2021) using question generation. Successfully, we promote the performance on multiple QA scenes and prove the expansibility of our model on different NLP tasks.

2 Related Work

Question generation is a promising task which has been well studied in many researches. Initially, rule-based or traditional machine learning methods are widely used in producing questions. Heilman and Smith (2010) adopt verb transformations and Berant et al. (2013) use semantic parsing to synthesize questions. Recently, deep learning techniques have given a further development of question generation. Du et al. (2017) use an LSTM (Hochreiter and Schmidhuber, 1997) model, and Sultan et al. (2020) adopt RoBERTa (Liu et al., 2019) model to generate questions.

At the same time, the strategies like multitask learning and self-training have been applied to improve the quality of generated questions. Zhou et al. (2019) and Ma et al. (2020) employ a multitask structure to generate coherent and fluent questions. Sachan and Xing (2018) and Rennie et al. (2020) adopt a self-training strategy to jointly learn to ask and answer questions. Alberti et al. (2019) use roundtrip consistency to filter out inconsistent results. Shinoda et al. (2021) generate noisy data and Sultan et al. (2020) employ nucleus sampling (Holtzman et al., 2020) to improve the diversity of questions. However, they mainly focus on only one quality aspect and most of them are based on predefined answers or original data.

As QG can produce meaningful questions, it has been widely used to promote other NLP tasks. Liu et al. (2020) use a constrained question rewriting way to generate new data for QA tasks. Wang et al. (2020) and Nan et al. (2021) check the faithfulness of summaries through answering generated questions. Pan et al. (2021) generate question-answer pairs and convert them for fact verification. Nevertheless, the researches above mainly produce each question independently and ignore the connections between questions.

As for generating a set of questions over a specific passage, Krishna and Iyyer (2019) propose a pipelined system to ask different levels of questions from general to specific. Lee et al. (2020) use conditional variational autoencoder to generate multiple robust questions for a given paragraph. Similar to us, Chai and Wan (2020) generate sequential and related questions under dual-graph interaction, but use ground-truth answers. To the best of our knowledge, we are the first to consecutively synthesize a series of connected question-answer pairs to understand an entire passage, with the comprehensive consideration of accuracy, coverage, and informativeness.

3 Multitask Framework

In our CQG strategy, the foundation is five various but unified tasks. The effects of these tasks are dynamically spread throughout our whole strategy. In section 4 we use them to compose four related methods to enhance different stages.

We first symbolically define the four key elements used in our work. S denotes the story from

S: *[Once upon a time in Greece, there lived a young man called Narcissus.]*^{stc_1} *[He lived in a small village on the sea and was famous in the land because he was quite handsome.]*^{stc_2} ...		
Q_1: What was the name of the young man?	A_1: Narcissus.	R_1: stc_1
Q_2: Where did he live?	A_2: A small village on the sea.	R_2: stc_2
Q_3: Was he famous in the land?	A_3: Yes.	R_3: stc_2
Q_4: Why?	A_4: Because he was quite handsome.	R_4: stc_2

Task	Input	Output
a	$Q_1A_1 \cdots Q_{n-1}A_{n-1} < sep > \ answer\ this : Q_n < sep > S$	A_n
q	$Q_1A_1 \cdots Q_{n-1}A_{n-1} < sep > \ question\ it : A_n < sep > S$	Q_n
$main$	$Q_1A_1 \cdots Q_{n-1}A_{n-1} < sep > \ pose\ pair : R_n < sep > S$	$Q_n ? A_n$
r	$Q_1A_1 \cdots Q_{n-1}A_{n-1} < sep > \ find\ rationale : Q_nA_n < sep > S$	R_n
h	$Q_1A_1 \cdots Q_nA_n < sep > \ generate\ history < sep >$	$\bigcup_{i=1}^{n} R_i$

Table 2: An ideal CQG example, where the questions are mutually connected and can cover diverse information to help understand the whole story. Also an example of data composition of our multitask generation framework, as well as the input and output in the nth generation step. In this example, the output of *Task* h is stc_1 when $n = 1$, and is $stc_1 stc_2$ when $n \geq 2$. "$\bigcup$" means coverage, or union set, with no overlap or replication.

which questions are produced; Q_n means the nth question and A_n is the answer; R_n is the corresponding rationale (always one sentence) based on which Q_n is generated. Since the Q-A pairs are generated dependently on previous questions, C_n denotes the context which composes of previous $n-1$ Q-A pairs and the story. Table 2 is an example. Then we define the main task and the four auxiliary tasks using the nth turn as follows:

Task $main$: $C_n + R_n \rightarrow Q_n + A_n$
Task a: $C_n + Q_n \rightarrow A_n$
Task q: $C_n + A_n \rightarrow Q_n$
Task r: $C_n + Q_n + A_n \rightarrow R_n$
Task h: $\sum_{i=1}^{n}(Q_i + A_i) \rightarrow \bigcup_{i=1}^{n} R_i$

In *Task* $main$, because we think the extractive answer is usually simple and it is inconsistent to get a Q-A in two steps, different from traditional methods, we input the context and rationale and output the question and answer simultaneously.

The design of *Task* a and *Task* q aims to guarantee that the generated question and answer are accurate: given the question we can get the answer and given the answer we can get the question. Here *Task* a follows traditional QA form. We do not input the rationale in *Task* q because previous Q-A pairs are included in the context, so if A_n is an accurate answer, the model should recognize the connection between the answer and the previous Q-A pairs, and restore the question easily.

Moreover, although we input the rationale in *Task* $main$, it does not necessarily imply that the question-answer pair is derived from it. So we design *Task* r ($C_n + Q_n + A_n \rightarrow R_n$) to verify that the model indeed uses the information in input rationale to get the question and answer. *Task* r helps the model to recognize the corresponding rationale, and then increase the coverage of a Q-A series, which means more events or more segments are precisely referred to.

Finally, to generate an informative and useful question, which means the knowledge it asks for does not overlap with previous ones, we consider that the more unseen information included in the Q-A pair, the better. We introduce the history of the context as the coverage of all previous rationales, which represents the total background information till the current Q-A turn. Therewith, we present *Task* h: $\sum_{i=1}^{n}(Q_i + A_i) \rightarrow \bigcup_{i=1}^{n} R_i$, which uses Q-A pairs to restore the history. "$\bigcup$" means cover, with no overlap or replication, and "+" means append or plus.

Both *Task* r and *Task* h use Q-A pairs to restore the context, but focus on coverage and informativeness differently. Specifically, a part of a story is covered means a question is asked based on it, but a informative question means it is non-trivial and important and contains no repetitive information. Also, in *Task* r we input the context, so the model only needs to locate the correct rationale, but in *Task* h, it has to generate the history completely based on Q-A pairs. Therefore in *Task* h, if the nth Q-A pair carries more unseen information, it will be easier to restore the history compared with a Q-A pair with repetitive or trivial information.

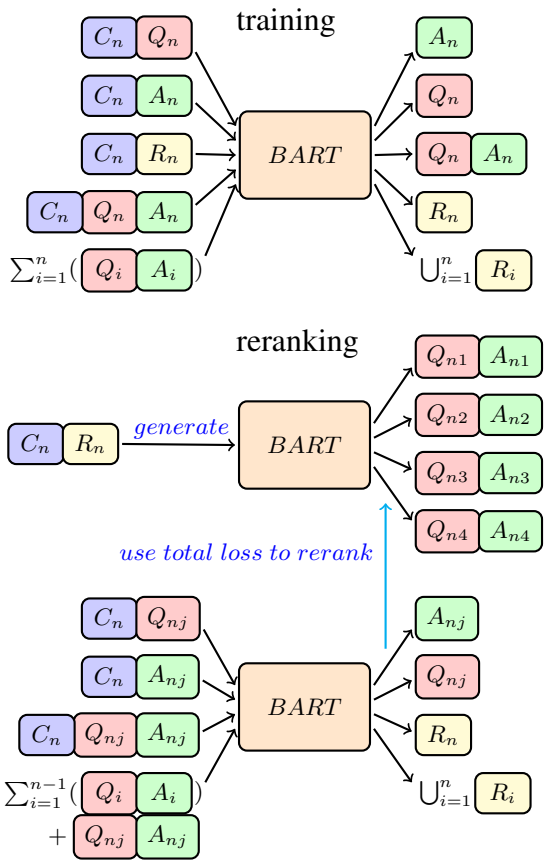

Figure 1: An overview of our dynamic multitask framework during joint training and self-reranking. One main task generates Q-A pairs and four auxiliary tasks generate other four CQG elements. In training, the five tasks are jointly trained in one model. In inference, the model uses the main task to generate candidates and then uses the auxiliary tasks to self-rerank them. We use the nth turn of a series of questions as an example and generate 4 candidates in inference. $j \in \{1, 2, 3, 4\}$.

4 Training and Inference

Based on the dynamic multitask framework, we jointly train a BART (Lewis et al., 2020) model. In inference, we use the main task to generate several candidates and self-rerank them using the auxiliary tasks. With the reranking losses, we design a formula to assess the information and automatically sample the rationales. Globally, we beam-search for the best Q-A series on sentence level.

4.1 Joint Training

We randomly shuffle the five kinds of training instances and use a BART model to jointly train the five tasks together. We also train the model to generate a "?" between a Q-A to split it, and adopt five hand-made prompts (Liu et al., 2021). Table 2 shows an example of our data structure. Given the Seq2Seq model parameterized by θ, the input sequence $\boldsymbol{x}$ with n tokens = $\{x_1, \cdots, x_n\}$ and label $\boldsymbol{y}$ with m tokens = $\{y_1, \cdots, y_m\}$, the generation probability and loss are as follows:

$$p(\boldsymbol{y}|\boldsymbol{x}, \theta) = \prod_{z=1}^{m} p(\boldsymbol{y}_z|\boldsymbol{y}_{<z}, \boldsymbol{x}, \theta) \quad (1)$$

$$loss(\boldsymbol{y}|\boldsymbol{x}, \theta) = -\frac{1}{m}\sum_{z=1}^{m} log\, p(\boldsymbol{y}_z|\boldsymbol{y}_{<z}, \boldsymbol{x}, \theta) \quad (2)$$

Through joint training we train a model to learn from different views and allow every task to benefit each other mutually. We also acquire the ability to do all five tasks in one model.

4.2 Self-Reranking

During the inference stage, through the main task we can obtain many candidate question-answer pairs using a decoding strategy like nucleus sampling. To select the best result, inspired by Shen et al. (2021), we employ these candidates to the same model to do *Task* a,q,r, and h, and then rank the candidates using the losses of the four auxiliary tasks. In another word, we use one model as both the generator and ranker. During reranking, the corresponding question and answer of the auxiliary tasks are those generated from *Task main*. Specifically, we multiply the four losses together as the reranking loss, as Eq.3, where the subscript i refers to different tasks.

$$loss_{rank}(\boldsymbol{y}|\boldsymbol{x}, \theta) = \prod_{i \in \{a,q,r,h\}} loss(\boldsymbol{y}_i|\boldsymbol{x}_i, \theta) \quad (3)$$

We consider the candidate with the lowest reranking loss as the one who excels in accuracy, coverage, and informativeness generally. This is inspired by the idea of evaluating generated text as text generation (Yuan et al., 2021). Through this strategy we also unify the form of training and reranking process and manage to do them in the same model. Figure 1 shows the structure of our multitask joint training and self-reranking.

4.3 Rationale Sampling

The aforementioned methods are useful to generate one good Q-A pair. Still, how to effectively generate consecutive questions on a passage remains unsettled. By default, we select every rationale as the next sentence of previous one. However, one rationale does not necessarily correspond to only one question, because a long informative sentence may be suitable for several Q-A pairs.

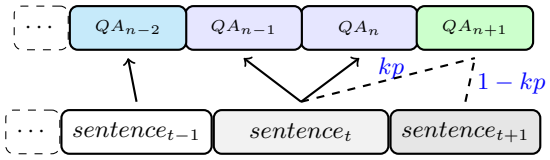

Figure 2: An example of rationale sampling, in which there is a probability of kp that R_{n+1} is $sentence_t$, and $1-kp$ it is $sentence_{t+1}$. Specifically in this example, n' is $n-2$, $R_{n'}$ is $sentence_{t-1}$, and $m_{n'}$ is the length of $\sum_{i=1}^{t-1} sentence_i$.

Hence, we propose the rationale sampling strategy, which introduces a probability that the next rationale keeps the same sentence as the current one, as Figure 2 shows. We use kp as the keeping probability. Then intuitively, we let kp be linearly related to the amount of information left in the current rationale. Traditionally, the information is hard to be calculated quantitatively. However, recall that we use the loss of *Task h* to measure the information of a Q-A series, so similarly, we design a loss to represent the rest information in current rationale. We want a higher loss to mean that less information of R_n is included in the Q-A series, and more information is still left in R_n.

Naturally, we first separate out the Q-A pairs on R_n. Given current step n, we find n', which is the most recent step where $R_{n'} \neq R_n$. Then, we use

$$loss(R_n | \sum_{i=n'+1}^{n} (Q_i + A_i) + \bigcup_{i=1}^{n'} R_i, \theta) \approx \frac{m_n loss_{h_n} - m_{n'} loss_{h_{n'}}}{m_n - m_{n'}} \triangleq a$$

to represent the rest information in R_n[3], which is the loss of using previous sentences and the Q-A pairs on R_n to restore R_n. Given our multitask framework, we use the ready-calculated losses of *Task h* to approximate this loss, without introducing more computation and complexity.

The approximation is a. Particularly if n is 1, a is $loss_{h_1}$. Empirically, we set the slope to be 0.2 and set a bound of 0-0.75. Finally, we get Eq.(4), and the average kp is 0.32 in the experiments, resulting in about 1.3 questions from one sentence.

$$kp = \begin{cases} 0, & 0.2a \leq 0 \\ 0.2a, & 0 < 0.2a < 0.75 \\ 0.75. & 0.2a \geq 0.75 \end{cases} \quad (4)$$

Besides, we also design other two rationale sampling strategies as in Appendix B.4, which shows that our strategy which bases on *Task h* to calculate information performs better than other handmade probability formulas.

[3] $m_n = \text{len}(\bigcup_{i=1}^{n} R_i)$. The details are in Appendix B.3

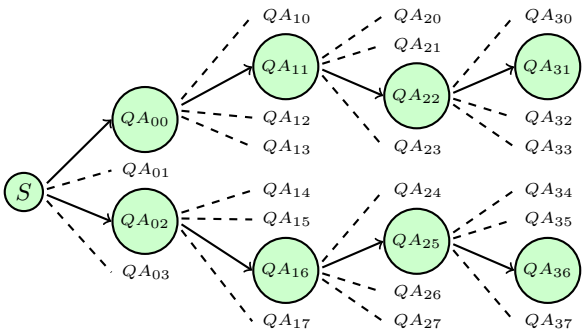

Figure 3: An overview of the sentence-level beam-search strategy. In this example each step the model generates 4 question-answer candidates and the sentence-level beam size is 2.

4.4 Sentence-Level Beam-Search

Although rationale sampling helps catch more information and improves flexibility, it brings about more uncertainty. The mutually dependent generation may also lead to deviation (Li et al., 2021). Thus, it is crucial to guide the flow direction in every step and ensure the quality of the whole series.

Naturally, inspired by traditional beam-search (token-level), we propose the sentence-level beam-search, as Figure 3 shows. Different from traditional beam-search, which generates a token in each search step, we generate a QA pair, and we adopt the reranking loss of each QA pair to take the place of the generation probability. Thus, in each step, we maintain several candidates with the lowest product of all previous reranking losses, which is calculated as Eq.5, where L is the final loss of our sentence-level beam-search method.

$$L(Q_1A_1 \cdots Q_nA_n | \boldsymbol{x}, \theta) = \prod_{j=1}^{n} loss_{rank_j} \quad (5)$$

To summarize, 3.2 to 3.4 are for inference. Practically, in each generation step, we first use previous results to do rationale sampling to locate the rationale, then generate some candidates and calculate the current reranking losses, and finally we use the total losses to sentence-level beam-search and keep several Q-A flows for the next step.

5 Experiments

5.1 Experimental Setup

We employ CoQA (Reddy et al., 2019) training set as our training data. CoQA is a large-scale dataset

for building Conversational Question Answering systems. The questions are conversational, and thus, every question after the first is dependent on the conversation history. The answers are free-form text with their corresponding rationales in the story. We expand the rationales to whole sentences and remove the questions with unknown answers. Finally, we get 7199 stories and each story has 15 turns of Q-A pairs on average. The training details and experiments are in Appendix A, where we also analyze the effect of joint training.

After training a model θ on CoQA, we evaluate our model by applying its question generation ability to two downstream tasks: data augmentation for QA and document-level NLI. Further, under the synthetic results on CoQA, we analyze their accuracy, coverage, and informativeness using human evaluations and a repeat-pose experiment.

5.2 Experiments to Augment QA Data

Data augmentation is one common way to employ generated questions and verify QG models. To augment QA dataset D, we (1) use θ to synthesize Q-A pairs D' on the training set of D; (2) train another BART model θ' on D' or $D + D'$ to answer questions[4]; (3) test θ' on the dev set of D.

Results on CoQA

First we test our strategy to augment CoQA dataset. The setting $Origin$ means the model θ' is trained on the original CoQA training set, and $Synth$ means it is trained with synthetic Q-A pairs. Inspired by Yuan et al. (2021), we additionally use the losses to measure the performance

In $Synth$, we conduct single q, two step, and single m as three baseline models, where single q means we use a single *Task q* model to ask questions based on the origin answers, like the traditional QG methods. Two step means we first extract an answer[5], then generate a question on it using the single *Task q* model. Single m is a *Task m* model, which generates Q-A pairs.

Joint train is a multitask jointly trained model. Based on joint train model, we further add the self-reranking method, using all four auxiliary tasks. Then on this joint train + rerank model, we conduct four ablation studies of auxiliary tasks.

Under joint train + rerank model, we also introduce other two conditions, independent and relay. By default, we generate the question series in an automatic way, which means every step the previous Q-A pairs are the Q-A pairs generated in previous steps. In independent setting, we let previous Q-A pairs be empty in all steps, which means the model generates every question like the first question, but when training QA model θ', we still input the previous QA pairs to align the data format with CoQA. In relay setting, the previous Q-A pairs of every synthetic instance are from CoQA training set, and the rationale is the ground-truth rationale sentence, which means the model inherits the Q-A flow from authentic CoQA's context.

Finally, still under joint train + rerank model, we add rationale sampling and sentence-level beam-search. Additionally, we merge the original training set with synthetic data to create the merging setting ($D + D'$). Note that RS and SBS are not suitable for independent or relay setting.

CoQA	**Bleu**	**Loss**	**F1**$_{qa}$
Origin			
Bart	38.52	0.777	78.54
Synth			
Single q	35.43/37.85	5.429/0.869	70.82/78.35
Two step	15.41/39.92	5.078/0.817	56.00/77.85
Single m	27.04/41.42	5.538/0.776	65.66/79.20
Joint train	26.97/38.92	5.613/0.765	65.90/80.11
+rerank	24.88/38.26	5.674/0.768	65.05/80.52
rerank a	25.31/38.03	5.612/0.764	63.71/80.23
rerank q	24.66/37.83	5.401/0.773	64.44/80.29
rerank r	24.03/38.05	5.487/0.768	63.73/80.18
rerank h	23.10/37.32	5.499/0.789	63.01/80.27
indep	20.38/39.03	5.490/0.783	56.54/78.29
relay	35.11/45.24	5.477/0.781	75.90/81.79
+RS	31.73/46.24	5.323/0.758	72.33/81.83
+SBS	32.01/**47.86**	5.431/**0.766**	72.49/**81.98**

Table 3: Results on CoQA dev set. In $Synth$, results without and with merging are separated by "/". In the middle are four ablation experiments of auxiliary tasks with Bart joint train+rerank. RS: rationale sampling. SBS: sentence-level beam-search.

Table 3 shows the results. The single q and two step model make relatively low scores when merged with original data, which means they generate relatively simple and low-quality questions. Using our one step Q-A pairs generation, in merging setting the single m model leads to higher scores even than single q, which based on origin answers. Joint train and reranking further improve

[4]Since our synthetic Q-A pairs are free-form, we still use BART to generate the answers on both CoQA and SQuAD.

[5]Use a BERT model to locate the start and end tokens.

the $F1_{qa}$ scores by 1.32 points. From the four ablation studies, it is not hard to see that every auxiliary task filters the results effectively, leading to 0.07 to 0.18 higher $F1_{qa}$ scores.

Comparing the independent settings with our model, we can see that our consecutive generation largely improves the quality of questions by 2.23 $F1_{qa}$ scores. Moreover, although the relay model based on the original Q-A flow truly gets better performance, when we add RS and SBS strategy to get our best model, the $F1_{qa}$ score is further increased by 1.46 points, and finally it outperforms relay generation by 0.19 points. It shows that the Q-A series searched by RS and SBS are more proper even than the ground-truth flow.

Results on SQuAD and more data

To check our QG ability on out-of-domain passages, we augment SQuAD (Rajpurkar et al., 2018) dataset using our best model trained on CoQA. We select the instances without unknown answers and with a story longer than 128 words. Since the questions in SQuAD are independent but also well-organized, we manually add previous Q-A pairs to align with CoQA.

To truly reveal the ability of our model, we employ it to synthesize more questions on a large number of unlabeled passages. We randomly collect 10000 Wikipedia passages whose lengths are from 100 to 500 words. Then we use our model trained on CoQA to generate questions on them, resulting in about 0.15 million Q-A pairs, which we use to augment both CoQA and SQuAD.

SQuAD	**Bleu**	**Loss**	**$F1_{qa}$**
Origin			
Bart	65.52	0.675	84.26
+preQA	**68.67**	0.625	85.32
Synth			
Ours	41.91/67.43	4.639/0.691	67.57/85.59
+Wiki	50.58/65.39	4.010/**0.630**	74.90/**85.88**
CoQA			
Ours	32.01/**47.86**	5.431/0.766	72.49/81.98
+Wiki	33.01/47.43	5.441/**0.758**	72.58/**82.21**
Large	52.36	0.521	87.90

Table 4: Results of out-of-domain generation on SQuAD dev set, and on Wikipedia passages. "Ours" means Bart joint train+rerank+RS+SBS. In *Synth*, results without and with merging are separated by "/".

Table 4 shows the results. We can see that the Q-A series indeed enhances question answering. It also indicates that even if our model is trained on different dataset, its synthesized questions still help a QA model gain 0.27 more $F1_{qa}$ points on SQuAD. With more Wikipedia questions, in both CoQA and SQuAD, we manage to further improve $F1_{qa}$ by 0.29 and 0.23 scores. It shows that our model performs well when transferring to another dataset and can augment the QA training sets with large-scale unlabeled data. Finally we adopt *large* model to get 87.90 $F1_{qa}$ points on CoQA.

5.3 Understand a Whole Passage (DocNLI)

To prove that our generated questions can really explore most information in an entire passage, we adopt our model for document-level NLI (DocNLI) task. Models are required to predict the relation (entailment or not) between a document-level premise and a hypothesis.

Traditionally, a model predicts the relation in a sequence classification way. However, given our ability to synthesize consecutive questions to understand a passage, we propose a zero-shot method to predict the relation based on question generating and answering. Since entailment requires the hypothesis to be derived from the premise, we first generate Q-A pairs given the hypothesis, and then answer these questions based on the premise. If we can get the same answers, we predict entailment. In detail, we (1) use θ to synthesize a series of Q-A pairs on the hypothesis; (2) use θ to answer Q on the premise, obtaining A'; (3) check the overlap ($F1_{qa}$) between A and A'. If the $F1_{qa}$ exceeds a given threshold, it is entailment.

To make sure that the passages are long enough to generate a series of Q-A pairs, we select the instances whose premise and hypothesis are 200 to 1000 words from all train, dev, and test set of DocNLI, to be our evaluation set. It is 1677 instances in all, and we averagely generate 15 turns of Q-A each instance with rationale sampling. We use 60 points of $F1_{qa}$ as the threshold of entailment.

Tabel 5 shows the results. $F1_{nli}$ is the harmonic mean of the precision and recall on the classification task. Impressively, using the zero-shot method, our best model surpasses the fine-tuned BERT model by 1.42 points of $F1_{nli}$ score. Among different QG settings, although two step model gets very low losses, its $F1_{nli}$ score is not very high, indicating that it generates relatively simple questions which cannot extract much information. Our one step model gets a lower

DocNLI	Loss	$F1_{qa}$	$F1_{nli}$
Finetune			
Bert	-	-	48.56
QG			
Two step	1.142/2.020	65.69/51.54	47.67
Single m	3.376/4.273	61.00/47.73	46.85
Joint train	3.223/4.119	63.32/49.56	46.90
+rerank	3.217/4.149	63.04/49.68	47.91
indep	2.811/3.857	63.90/49.18	47.88
+RS	2.633/3.601	65.98/50.99	49.88
+SBS	2.376/3.353	66.19/51.19	**49.98**

Table 5: Results of DocNLI task. Finetune is a BERT-base model fine-tuned on about 0.8 million other DocNLI instances. When using our zero-shot method, QA results of entailment and not entailment are separated by "/". We use different models for QG, and the QA model is the same as our best model θ.

$F1_{nli}$ score initially but with the joint training and reranking strategy, it improves the score by 0.98 points. Moreover, we can see clearly that the RS and SBS strategies improve the result significantly by 2.10 $F1_{nli}$ scores. They also manage to enlarge the discrimination between entailment and not entailment. It suggests that our consecutive generation strategy really produces question-answer pairs with most of the information in a passage, which can help understand the passage effectively.

5.4 Analyses

Accuracy and Coverage (*Task* a, q and r)

Here we conduct two human evaluations, to prove that our strategy improves Q-A accuracy and story coverage, which are the effects of *Task* a, q and *Task* r. Since the coverage requires the model to ask for more points of a passage, we use the question-rationale consistency (accuracy of rationale) to reflect it. This is because all sentences are asked at least once, and rationale sampling further guarantees the rationales to be well-distributed, so if the rationales are all precisely questioned, the coverage should be as well satisfactory.

We randomly collect 10% stories from CoQA dev set and use different methods to generate Q-A pairs. We, the authors, then manually measure whether every question is correctly asked and answered and whether every question-answer pair is derived from its corresponding rationale.

Table 6 clearly shows that multitask joint training and reranking and sentence-level beam-search increase the accuracy of Q-A by 6.52 % and rationale by 5.39 %. Thus, we can say that our strategy, main due to *Task* a, q and *Task* r, helps generate questions more correctly and locate the rationale more precisely, leading to higher Q-A accuracy and coverage in a series of questions.

Acc of	Ours	-SBS	-Rerank	-Joint train
Q-A pair	**94.85**	92.71	90.32	88.33
rationale	**95.65**	93.89	90.97	90.26

Table 6: Human evaluations of accuracy of Q-A and rationale. We do not ablate RS here because it is not relevant here and will make the data unaligned.

Informativeness (*Task* h)

To evaluate the ability to utilize information in a rationale, we present the repeat-pose experiment on CoQA. It is adapted from relay setting, and requires the model to pose another question based on the same rationale and same context as the original question. In other words, the model has to "squeeze" more information from the same rationale, so the key is whether *Task* h can rank the informativeness of each candidate precisely.

CoQA	Bleu	Loss	$F1_{qa}$
Bart joint train relay w/o rerank	41.01	0.737	81.21
Bart joint train repeat w/o rerank	41.97	0.741	81.28
Bart joint train repeat w/ rerank	**43.40**	**0.708**	**81.57**

Table 7: Results of the repeat-pose experiment. Synthetic data are merged with the original training set.

Table 7 shows the results, which demonstrate that repeat-pose with self-reranking strategy further improves the $F1_{qa}$ scores by 0.36 points, indicating that *Task* h indeed helps select the more informative question-answer pairs.

6 Conclusion

In this paper, we propose the consecutive question generation task, which synthesizes mutually connected question-answer pairs to fully explore the information in a passage. By constructing a novel multitask framework with one main task and four unified auxiliary tasks, we generate optimum Q-A series using four sub-methods, which help "generate good questions" as well as "find worth-asking information". With extensive experiments, we prove that our model is able to generate high-quality Q-A pairs to understand a whole passage and has the power to benefit various NLP tasks.

References

Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. 2019. Synthetic QA corpora generation with roundtrip consistency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6168–6173, Florence, Italy. Association for Computational Linguistics.

Max Bartolo, Tristan Thrush, Robin Jia, Sebastian Riedel, Pontus Stenetorp, and Douwe Kiela. 2021. Improving question answering model robustness with synthetic adversarial data generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8830–8848, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

Antoine Bordes and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. *ArXiv*, abs/1605.07683.

Zi Chai and Xiaojun Wan. 2020. Learning to ask more: Semi-autoregressive sequential question generation under dual-graph interaction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 225–237.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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

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, Copenhagen, Denmark. Association for Computational Linguistics.

Esin Durmus, He He, and Mona Diab. 2020. FEQA: A question answering evaluation framework for faithfulness assessment in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5055–5070, Online. Association for Computational Linguistics.

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

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9:1735–1780.

Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. *ArXiv*, abs/1904.09751.

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

Payal Khullar, Konigari Rachna, Mukul Hase, and Manish Shrivastava. 2018. Automatic question generation using relative pronouns and adverbs. In *Proceedings of ACL 2018, Student Research Workshop*, pages 153–158, Melbourne, Australia. Association for Computational Linguistics.

Kalpesh Krishna and Mohit Iyyer. 2019. Generating question-answer hierarchies. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2321–2334, Florence, Italy. Association for Computational Linguistics.

Dong Bok Lee, Seanie Lee, Woo Tae Jeong, Donghwan Kim, and Sung Ju Hwang. 2020. Generating diverse and consistent QA pairs from contexts with information-maximizing hierarchical conditional VAEs. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 208–224, Online. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Patrick Lewis, Yuxiang Wu, Linqing Liu, Pasquale Minervini, Heinrich Kuttler, Aleksandra Piktus, Pontus Stenetorp, and Sebastian Riedel. 2021. Paq: 65 million probably-asked questions and what you can do with them. *Transactions of the Association for Computational Linguistics*, 9:1098–1115.

Huihan Li, Tianyu Gao, Manan Goenka, and Danqi Chen. 2021. Ditch the gold standard: Re-evaluating conversational question answering. *ArXiv*, abs/2112.08812.

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, Sofia, Bulgaria. Association for Computational Linguistics.

Dayiheng Liu, Yeyun Gong, Jie Fu, Yu Yan, Jiusheng Chen, Jiancheng Lv, Nan Duan, and Ming Zhou. 2020. Tell me how to ask again: Question data augmentation with controllable rewriting in continuous space. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5798–5810, Online. Association for Computational Linguistics.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ArXiv*, abs/2107.13586.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Xiyao Ma, Qile Zhu, Yanlin Zhou, Xiaolin Li, and Dapeng Oliver Wu. 2020. Improving question generation with sentence-level semantic matching and answer position inferring. *ArXiv*, abs/1912.00879.

Feng Nan, Cicero Nogueira dos Santos, Henghui Zhu, Patrick Ng, Kathleen McKeown, Ramesh Nallapati, Dejiao Zhang, Zhiguo Wang, Andrew O. Arnold, and Bing Xiang. 2021. Improving factual consistency of abstractive summarization via question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6881–6894, Online. Association for Computational Linguistics.

Liangming Pan, Wenhu Chen, Wenhan Xiong, Min-Yen Kan, and William Yang Wang. 2021. Zero-shot fact verification by claim generation. In *ACL/IJCNLP*.

Raul Puri, Ryan Spring, Mohammad Shoeybi, Mostofa Patwary, and Bryan Catanzaro. 2020. Training question answering models from synthetic data. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5811–5826, Online. Association for Computational Linguistics.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

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.

Steven Rennie, Etienne Marcheret, Neil Mallinar, David Nahamoo, and Vaibhava Goel. 2020. Unsupervised adaptation of question answering systems via generative self-training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1148–1157, Online. Association for Computational Linguistics.

Mrinmaya Sachan and Eric Xing. 2018. Self-training for jointly learning to ask and answer questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 629–640, New Orleans, Louisiana. Association for Computational Linguistics.

Jianhao Shen, Yichun Yin, Lin Li, Lifeng Shang, Xin Jiang, Ming Zhang, and Qun Liu. 2021. Generate & rank: A multi-task framework for math word problems. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2269–2279, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Kazutoshi Shinoda, Saku Sugawara, and Akiko Aizawa. 2021. Improving the robustness of QA models to challenge sets with variational question-answer pair generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: Student Research Workshop*, pages 197–214, Online. Association for Computational Linguistics.

Md Arafat Sultan, Shubham Chandel, Ramón Fernandez Astudillo, and Vittorio Castelli. 2020. On the importance of diversity in question generation for QA. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5651–5656, Online. Association for Computational Linguistics.

Kai Sun, Dian Yu, Dong Yu, and Claire Cardie. 2019. Improving machine reading comprehension with general reading strategies. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2633–2643, Minneapolis, Minnesota. Association for Computational Linguistics.

Xingwu Sun, Jing Liu, Yajuan Lyu, Wei He, Yanjun Ma, and Shi Wang. 2018. Answer-focused and

position-aware neural question generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3930–3939, Brussels, Belgium. Association for Computational Linguistics.

Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *ArXiv*, abs/1706.03762.

Alex Wang, Kyunghyun Cho, and Mike Lewis. 2020. Asking and answering questions to evaluate the factual consistency of summaries. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5008–5020, Online. Association for Computational Linguistics.

Siyuan Wang, Zhongyu Wei, Zhihao Fan, Yang Liu, and Xuanjing Huang. 2019. A multi-agent communication framework for question-worthy phrase extraction and question generation. In *AAAI*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2020. Transformers: State-of-the-art natural language processing. In *EMNLP*.

Wenpeng Yin, Dragomir Radev, and Caiming Xiong. 2021. DocNLI: A large-scale dataset for document-level natural language inference. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4913–4922, Online. Association for Computational Linguistics.

Weizhe Yuan, Graham Neubig, and Pengfei Liu. 2021. Bartscore: Evaluating generated text as text generation. *ArXiv*, abs/2106.11520.

Wenjie Zhou, Minghua Zhang, and Yunfang Wu. 2019. Multi-task learning with language modeling for question generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3394–3399, Hong Kong, China. Association for Computational Linguistics.

A Implementation and Training Details

We use PyTorch to implement our models. We acquire the pre-trained BART model[6] from the Transformers library (Wolf et al., 2020).

During training, we set the batch size to 64 and learning rate to 1e-5. The maximum input length is 1024. In inference, we use beam-search with beam size 4 to generate answers for QA. Following Sultan et al. (2020), we use nucleus sampling with top-k(k=50) and top-p(p=0.95) to generate question-answer pairs. We averagely return 4 candidates each step and set sentence-level beam size to 4, which means in our best model, every step we select 4 out of 16 candidate Q-A flows. The models we use are *base* size.

After training we evaluate the losses of five tasks on CoQA dev set, and the $F1_{qa}$ scores using *Task a*. Table 8 shows the results with different training settings. We can see that joint training improves the performance on four out of five tasks, suggesting that different tasks benefit each other effectively. Prompts also enhance the Q-A ability and decrease the losses on three out of five tasks.

CoQA	Ours	w/o Prompts	w/o Joint
Loss a	**0.767**	0.771	0.777
Loss q	**1.364**	1.370	1.377
Loss m	**1.372**	1.378	1.388
Loss r	0.062	**0.058**	0.068
Loss h	2.554	2.543	**2.536**
$F1_{qa}$ a	**80.60**	80.07	78.54

Table 8: Losses and $F1_{qa}$ scores on CoQA dev set using different training method.

During reranking, the scales of different losses are also not far from Table 8.

B Supplementary Analyses

B.1 Beam-Search or Nucleus Sampling

As argued in (Sultan et al., 2020), nucleus sampling leads to higher diversity and is better than beam-search in QG. To verify that, we train two sets of models on different tasks with full strategies. We adopt beam-search with size 4 and nucleus sampling with top-k(k=50) and top-p(p=0.95). Table 9 shows that nucleus sampling truly gains better results than beam-search.

Tasks	Beam-Search	Nucleus Sampling
CoQA	0.765/81.60	0.766/**81.98**
SQuAD	0.679/85.51	0.691/**85.59**
DocNLI	2.380/49.33	2.376/**49.98**

Table 9: Results using beam-search or nucleus sampling.

B.2 Efficiency Analysis

When training the multitask model, we jointly train five tasks in one model, so the efficiency of

[6] https://huggingface.co/facebook/bart-base

our strategy is an inevitable topic. Here in Figure 4, we demonstrate the training curves of *Task* a and m using single model and multitask model.

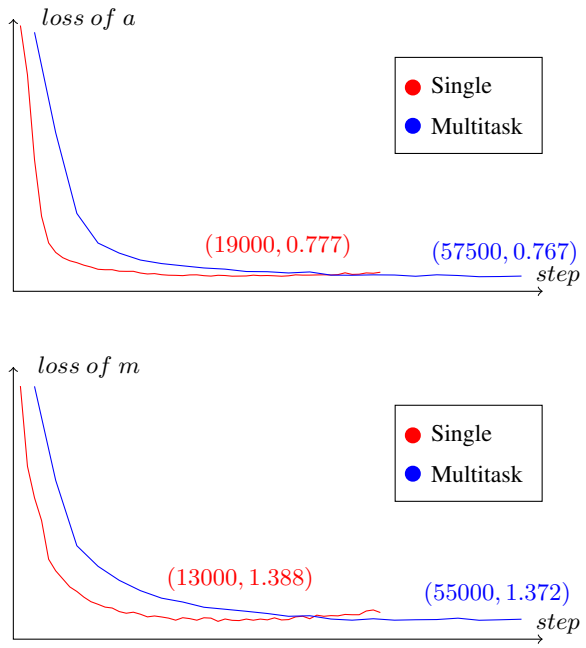

Figure 4: The training curves of *Task* a and m using single model and multitask model. The optimum points are marked in the figures. Note that our batch size is 64.

We can clearly see that the convergence speed of multitask model is not five times slower than the single model. In fact, it only takes about three times of steps in *Task* a and four times in *Task* m, for our multitask model to meet the optimum point compared with the single model. Also, the initial convergence speed in the first few steps of the single model is only about twice as fast as the joint model. Thus, in training we can say that the five tasks mutually benefit each other. In inference our multitask model takes about five times as long to generate a question.

B.3 Mathematically Analysis of Rationale Sampling

Although the intuition of our rationale sampling is to use previous sentences and the Q-A pairs on R_n to restore R_n, $\sum_{i=n'+1}^{n}(Q_i+A_i)$ is dependent on and logically connected with $\sum_{i=1}^{n'}(Q_i+A_i)$. Also, since the information of $\sum_{i=1}^{n'}(Q_i+A_i)$ is totally contained in $\bigcup_{i=1}^{n'} R_i$, we might as well do the following transformation.

$$loss(R_n|\sum_{i=n'+1}^{n}(Q_i+A_i)+\bigcup_{i=1}^{n'}R_i,\theta)$$
$$\approx loss(R_n|\sum_{i=1}^{n}(Q_i+A_i)+\bigcup_{i=1}^{n'}R_i,\theta).$$

Also, since the information of $\sum_{i=n'+1}^{n}(Q_i+A_i)$ contribute not much to generate $\bigcup_{i=1}^{n'} R_i$, we can say that

$$p(\bigcup_{i=1}^{n'}R_i|\sum_{i=1}^{n}(Q_i+A_i),\theta)$$
$$\approx p(\bigcup_{i=1}^{n'}R_i|\sum_{i=1}^{n'}(Q_i+A_i,\theta).$$

Then,

$$\begin{aligned}
&loss(R_n|\sum_{i=n'+1}^{n}(Q_i+A_i)+\bigcup_{i=1}^{n'}R_i,\theta)\\
&\approx loss(R_n|\sum_{i=1}^{n}(Q_i+A_i)+\bigcup_{i=1}^{n'}R_i,\theta)\\
&=-\frac{log\,p(R_n|\sum_{i=1}^{n}(Q_i+A_i)+\bigcup_{i=1}^{n'}R_i,\theta)}{m_n-m_{n'}}\\
&=-\frac{1}{m_n-m_{n'}}[\\
&\quad log\,p(R_n|\sum_{i=1}^{n}(Q_i+A_i)+\bigcup_{i=1}^{n'}R_i,\theta)\\
&+log\,p(\bigcup_{i=1}^{n'}R_i|\sum_{i=1}^{n}(Q_i+A_i),\theta)\\
&-log\,p(\bigcup_{i=1}^{n'}R_i|\sum_{i=1}^{n}(Q_i+A_i),\theta)]\\
&=-\frac{1}{m_n-m_{n'}}[log\,p(\bigcup_{i=1}^{n}R_i|\sum_{i=1}^{n}(Q_i+A_i),\theta)\\
&-log\,p(\bigcup_{i=1}^{n'}R_i|\sum_{i=1}^{n}(Q_i+A_i),\theta)]\ (use\ Eq.2)\\
&\approx-\frac{1}{m_n-m_{n'}}[log\,p(\bigcup_{i=1}^{n}R_i|\sum_{i=1}^{n}(Q_i+A_i),\theta)\\
&-log\,p(\bigcup_{i=1}^{n'}R_i|\sum_{i=1}^{n'}(Q_i+A_i),\theta)]\\
&=\frac{1}{m_n-m_{n'}}(m_n loss_{h_n}-m_{n'}loss_{h_{n'}})\triangleq a.
\end{aligned}$$

B.4 Other Rationale Sampling Strategies

Besides the rationale sampling strategy in 4.3, we also conduct two other versions. The first one is a constant function with a value of 0.3, as Eq.6. In the second version, we use the length of each rationale on behalf of its amount of information. We let x mean the ratio between the current rationale length and the story length and make kp linear related to x. Empirically, we set the slope to 3 and an upper bound of 0.75, as Eq.7.

$$kp = 0.3. \tag{6}$$

$$kp = \begin{cases} 3x, & 0 \leq x \leq 0.25 \\ 0.75, & 0.25 < x \leq 1 \end{cases} \tag{7}$$

Tasks	Eq.6	Eq.7	Ours
CoQA	0.772/81.62	0.762/81.88	0.766/**81.98**
SQuAD	0.660/85.43	0.651/**85.61**	0.691/85.59
DocNLI	2.382/49.12	2.375/49.88	2.376/**49.98**

Table 10: Results using different rationale sampling strategies.

Using these three rationale sampling methods, we train three sets of models on different tasks with full strategies. The results are in Table 10. We can see that the dynamic probability is more suitable than the constant value. Also, our strategy based on auxiliary *Task h* performs better than that based on sentence length. Specifically, it gets 0.1 points higher on CoQA and DocNLI and gets almost the same score on SQuAD.

C Example Analysis

In this paper, we propose the consecutive question generation strategy which mainly focuses on the accuracy, coverage, and informativeness of a series of Q-A pairs generated on a whole passage. Here we further analyze the improvements of our model with a specific example. In Table C, using one passage in CoQA dev set, we present the synthesized questions produced by our model, compared with Two step, Single m, Joint train + rerank independent model, and the original data.

From the example, we can see that the original data contains 20 turns of Q-A pair, but the Q-A 17 to Q-A 20 are out of order. Our model generates 15 Q-A pairs, which is 4 turns more than other models, thanks to the rationale sampling strategy. For instance, the QA 14 and QA 15 of our model both focus on the last sentence, yet ask for two totally different information, which means our strategy really helps explore more details in a passage.

In the two step and single m model, QA 4 is inconsistent and QA 8 is grammatical erroneous, which is not accurate.

Meanwhile, since single m model does not sample rationales and asks questions sentence by sentence, QA 5 should focus on the "The girls dog $\cdots$ up ahead" rationale. However, it asks a question still based on the previous sentence, which means although we input the rationale, it gets the wrong information and asks twice about the fact that the girl was scared to go ahead, and misses a question about the dog's behavior. This is why we relate rationale accuracy to coverage and regard *Task r* as an important task.

In the joint train + rerank independent model, because the questions are generated independently and *Task h* cannot be helpful, QA 2 asks for the same answer as QA 1, which provide little information and where other models properly ask about the location they travel to. It proves that our mutually connected consecutive question generation is beneficial.

Additionally, the question series of two step and joint train + rerank independent model lack fluency. The pre-generated answers are often too long to be proper answers, and the independent Q-A pairs are too stiff and crude, mainly because of the missing of connections. Finally, among the examples we can say with confidence, our model generates the best question-answer series and explore the passage most appropriately.

S: *This is the story of a young girl and her dog. The young girl and her dog set out a trip into the woods one day. Upon entering the woods the girl and her dog found that the woods were dark and cold. The girl was a little scared and was thinking of turning back, but yet they went on. The girl's dog was acting very interested in what was in the bushes up ahead. To both the girl and the dog's surprise, there was a small brown bear resting in the bushes. The bear was not surprised and did not seem at all interested in the girl and her dog. The bear looked up at the girl and it was almost as if he was smiling at her. He then rested his head on his bear paws and went back to sleep. The girl and the dog kept walking and finally made it out of the woods. To this day the girl does not know why the bear was so friendly and to this day she has never told anyone about the meeting with the bear in the woods.*

Q_1: Who is the story about? A_1: Young girl and her dog.
Q_1: Who is this story about? A_1: Young girl and her dog.
Q_1: Who is the story about? A_1: Young girl and her dog.
Q_1: Who is the story about? A_1: Young girl and her dog.
Q_1: What is the story about? A_1: A girl and a dog.

Q_2: What did they do? A_2: Set out trip into woods.
Q_2: Where did they go? A_2: Into woods.
Q_2: Where did they go? A_2: Into woods.
Q_2: Who set out on a trip? A_2: Young girl and her dog. (Not informative)
Q_2: What were they doing? A_2: Set on on a trip.

Q_3: When? A_3: One day.
Q_3: What were the woods like? A_3: Dark and cold.
Q_3: Was it warm there? A_3: No.
Q_3: Were the woods dark and cold? A_3: Yes.
Q_3: Where? A_3: The woods.

Q_4: Did they find the woods warm? A_4: No.
Q_4: Why did they go on? A_4: Girl was scared and was thinking of turning back.(Not accurate)
Q_4: Was the girl scared? A_4: Yes.
Q_4: Who was thinking of turning back? A_4: Girl.
Q_4: How did the girl feel? A_4: Scared.

Q_5: How did the girl feel? A_5: Scared.
Q_5: Why did they go on? A_5: Dog was acting very interested in what was in bushes up ahead.
Q_5: What was she thinking of? A_5: Turning back.(Lack of coverage)
Q_5: Who was interested in what was in the bushes? A_5: Dog.
Q_5: How about the dog? A_5: He wasn't.

Q_6: What was she thinking of? A_6: Turning back.
Q_6: What was in the bushes? A_6: Small brown bear.
Q_6: What was in the bushes? A_6: Bear.
Q_6: What was in the bushes? A_6: Small brown bear.
Q_6: How did he feel? A_6: He was interested.

Q_7: What was the dog doing? A_7: Acting very interested in what was in bushes up ahead.
Q_7: Did the bear seem surprised? A_7: No.
Q_7: Did it seem interested? A_7: No.
Q_7: What did the bear not seem at all interested in? A_7: Girl and her dog.

Q_7: In what? A_7: What was in the bushes.

Q_8: What was it? A_8: Small brown bear.
Q_8: What did the bear do? A_8: He looked up at girl and it was almost as if he was smiling at her.
Q_8: What was it? A_8: Almost as if he was smiling at her. (Not accurate)
Q_8: How did the bear look at the girl? A_8: it was almost as if he was smiling at.
Q_8: What was it? A_8: A bear.

Q_9: What was it doing? A_9: Resting in bushes.
Q_9: What did he do next? A_9: He rested his head on his bear paws and went back to sleep.
Q_9: What did he do next? A_9: Rested his head on his bear paws.
Q_9: What did the bear do then? A_9: Rested his head on his bear paws and went back to sleep.
Q_9: What did it do? A_9: Rested in the bushes.

Q_{10}: Did the bear seem upset? A_{10}: No.
Q_{10}: What happened next? A_{10}: Girl and dog kept walking and finally made it out of woods.
Q_{10}: Did they make it out of the woods? A_{10}: Yes.
Q_{10}: Did the girl make it out of the woods? A_{10}: Yes.
Q_{10}: Did it notice the two? A_{10}: Not really.

Q_{11}: What did he do? A_{11}: Looked up at girl and it was almost as if he was smiling.
Q_{11}: What does the girl think about the meeting with the bear?
A_{11}: Girl does not know why bear was so friendly and to this day she has never told anyone.
Q_{11}: Did the girl know why the bear was so friendly? A_{11}: No.
Q_{11}: What did the young girl not know? A_{11}: Why bear was so friendly.
Q_{11}: How did the girl and the dog feel? A_{11}: Surprised.

Q_{12}: What did he do next? A_{12}: Rested his head on his bear paws and went back to sleep.
Q_{12}: How did the bear react? A_{12}: Not surprised.

Q_{13}: Did they make it out of the woods? A_{13}: Yes.
Q_{13}: What did he do? A_{13}: Looked at the girl.

Q_{14}: Did the girl know why the bear was so friendly? A_{14}: No.
Q_{14}: Was he mean? A_{14}: He smiled.

Q_{15}: Did the girl tell anyone? A_{15}: No.
Q_{15}: Does she know why? A_{15}: No.

Q_{16}: Who did she tell? A_{16}: No one.

Q_{17}: Was the woods open and light? A_{17}: No.

Q_{18}: What was it like? A_{18}: Dark and cold.

Q_{19}: Was she thinking of turning back? A_{19}: Yes.

Q_{20}: Did she? A_{20}: No.