Neural Question Generation using Interrogative Phrases

Yuichi Sasazawa **Sho Takase** **Naoaki Okazaki**
Tokyo Institute of Technology
{yuichi.sasazawa, sho.takase} at nlp.c.titech.ac.jp
okazaki at c.titech.ac.jp

Abstract

Question Generation (QG) is the task of generating questions from a given passage. One of the key requirements of QG is to generate a question such that it results in a target answer. Previous works used a target answer to obtain a desired question. However, we also want to specify how to ask questions and improve the quality of generated questions. In this study, we explore the use of interrogative phrases as additional sources to control QG. By providing interrogative phrases, we expect that QG can generate a more reliable sequence of words subsequent to an interrogative phrase. We present a baseline sequence-to-sequence model with the attention, copy, and coverage mechanisms, and show that the simple baseline achieves state-of-the-art performance. The experiments demonstrate that interrogative phrases contribute to improving the performance of QG. In addition, we report the superiority of using interrogative phrases in human evaluation. Finally, we show that a question answering system can provide target answers more correctly when the questions are generated with interrogative phrases.[1]

1 Introduction

Question Generation (QG) is the task of generating questions from a given passage. It has several applications: (1) In the area of the education, QG can help to generate questions for reading comprehension materials (Heilman and Smith, 2010). (2) QG can aid development of conversational chatbots, which ask questions (Mostafazadeh et al., 2016). (3) QG is useful for development of question answering datasets (Duan et al., 2017; Tang et al., 2018).

One of the key requirements of QG is to generate a question such that it asks a target answer. For example, the sentence "Bob went to the airport yesterday." can have various candidate questions such as "When did Bob go to the airport?," "Where did Bob go yesterday?," and "Who went to the airport yesterday?" It is necessary for QG to specify a desired question from among multiple possibilities. Du et al. (2017) and Chali et al. (2018) used the sequence-to-sequence model (Bahdanau et al., 2015), which takes only a passage as the input. Thus, only one question is generated from multiple possibilities at random. Most recent studies tried to generate a desired question using a target answer as the input. Zhou et al. (2017) and Song et al. (2018) incorporated the target answer using the answer position feature. Kim et al. (2018) separated target answer words from the original passages to address the problem of many generated questions including the target answer words.

However, we also want to specify how questions should be asked and improve the quality of the generated questions. The existing method sometimes generates inappropriate questions whose interrogative phrases do not match the target answers. For example, we specify a target answer, "in 1920," but an interrogative phrase of the generated question is "how much." Sun et al. (2018) proposed the answer-focused model to address this problem. Heilman and Smith. (2010) extracted target answers from passages and automatically converted them into interrogative phrases by a sequence of general rules.

In this study, we explore the use of interrogative phrases as additional sources to control QG. Using appropriate interrogative phrases that match the target answers is important for generating questions. Unlike Heilman and Smith (2010) or Sun et al. (2018), we directly input the correct interrogative phrases. Selecting correct interrogative

[1]Our code is available at https://github.com/WERimagin/NQG_Interrogative_Phrases.

Proceedings of The 12th International Conference on Natural Language Generation, pages 106–111,
Tokyo, Japan, 28 Oct - 1 Nov, 2019.

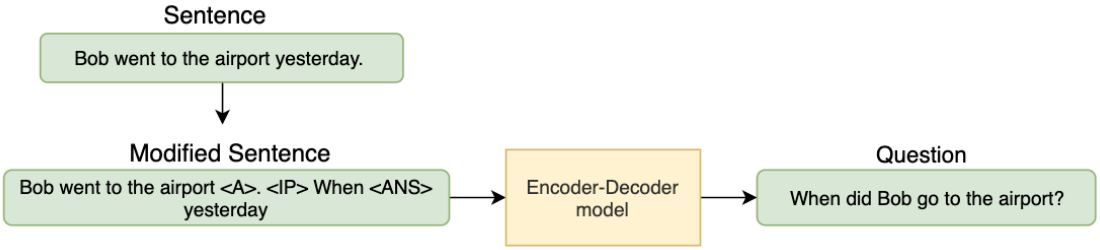

Figure 1: Overview of the proposed method. We concatenate an answer and interrogative phrase at the end of the passage using the special tokens <IP> and <ANS>. Also, we replace the answer phrase with the special token <A>. We use this modified passage as the input and generate a question.

phrases manually is not so difficult; thus, using the correct interrogative phrases as inputs is an expected experimental setting. We investigate to what extent using proper interrogative phrases as inputs contributes to the quality of the questions. To the best of our knowledge, this is the first study that studies the use of interrogative phrases as inputs to improve the quality of questions in QG.

We test our method on the SQuAD dataset (Rajpurkar et al., 2016). We demonstrate that the use of interrogative phrases contributes to improving the performance of QG by automatic metrics and human evaluation. We also conduct a question answering experiment. We show that it is easier for a question answering system to provide the correct target answers when questions are generated using interrogative phrases.

2 Method

In this work, we generate a question from a passage, a target answer, and an interrogative phrase. An outline of our method is shown in Figure 1.

We use the Encoder-Decoder model with an attention mechanism (Bahdanau et al., 2015) for generating questions. In order to obtain a question that depends on a passage, a target answer and an interrogative phrase, we concatenate the target answer and the interrogative phrase at the end of the passage using the special tokens <IP> and <ANS>. This treatment can inform the model about the target answer and the interrogative phrase that should be considered in the generated questions. In addition, we replace the span of the target answer in the passage with the special token <A>. This treatment helps the model to focus on the words present near the answer in the passage when generating a question.

Because a question is expected to use the words appearing in the passage, we incorporate the copy mechanism (See et al., 2017) into the model. By copying the words in the passage while generating the question, this mechanism reduces the risk of generating words not included in the passage. We also explore the use of the coverage mechanism (Tu et al., 2016) to avoid generating an inappropriate question. This mechanism can also prevent the model from generating the same word repeatedly.

Although this model is simpler than other QG models, it achieves the surprisingly good performance on the SQuAD v1.1 dataset, as described in Sections 3.4 and 3.5.

3 Experiments

3.1 Dataset

We use the SQuAD v1.1 dataset (Rajpurkar et al., 2016) in this study. The SQuAD dataset is a question answering dataset containing 107,785 questions with 536 articles. We lowercase all the data. We extract a sentence containing an answer phrase. Then, we use it as the input passage. If the answer phrase spans multiple sentences, we extract these sentencese and use the concatenation of them as the input passage. We use Stanford CoreNLP (Manning et al., 2014) to extract the interrogative phrases from the questions. If the CoreNLP detects multiple words as the interrogative phrases (e.g., "how many", "in what year" or "what country"), we use all the word as the interrogative phrase. We remove the sentence-question pairs whose questions do not contain an interrogative phrase. For instance, Yes/No questions (e.g., "Did you go to the school?") or questions whose interrogative phrase is located in the middle of the question, which Stanford CoreNLP cannot detect accurately (e.g., "Bob went to the airport how many times?"). We remove passage-question pairs that have no content (non-stop) word in common.

These pairs are unsuitable for the QG dataset because the passage is not related to the question. We thus obtain 74,863, 4,658, and 4,658 pairs for training, development, and test, respectively.

3.2 Experiment setting

We use the OpenNMT system (Klein et al., 2017). We retain 45,000 of the most frequent words on the source side and 28,000 of the most frequent words on the target side. All the other words are replaced by the <UNK> token. We use 300 dimensions pretrained glove.840B.300d (Pennington et al., 2014) embeddings for initialization, and we fix them during training. We use 2-layer long short-term memory in both the encoder and the decoder. The size of the hidden state is 600. The dropout rate is 0.3. We use stochastic gradient descent for optimization. The initial learning rate is 1.0. We halve the learning rate every 2,500 steps from 20,000 steps onwards. We finish the training at 50,000 steps. We adopt the model that achieves the highest accuracy in the development set.

During inference, we conduct the beam search with a beam width of 2. Decoding stops when the model generates the <EOS> token. All hyperparameters are tuned using the development set, and the results are reported using the test set.

We use two models for comparisons.

ASs2s This model is the state-of-the-art of the neural QG model for the SQuAD dataset (Kim et al., 2018). Based on the Encoder-Decoder architecture, this model incorporates a target answer using answer-separated seq2seq and the keyword-net module. This model need to use a target answer. We compare the accuracy of using interrogative phrases as inputs. When we use an interrogative phrase as the input, we concatenate it at the end of the passage using the special token <IP>.

Seq2Seq This is our baseline model. We use the RNN-based Encoder-Decoder model with the attention, copy, and coverage mechanisms. We compare the accuracy of using interrogative phrases or target answers as inputs. When we use an interrogative phrase as the input, we concatenate it at the end of the passage using the special token <IP>. When we use a target answer as the input, we concatenate it at the end of the passage using the special token <ANS> and replace the span of the target answer in the passage with special token <A>.

To check the contributions of the components in our model, we conduct ablation tests. We remove each component from the Seq2Seq + Answer + Interrogative method.

-Copy: Without the copy mechanism.

-Coverage: Without the coverage mechanism.

-Answer: Without concatenating the target answer at the end of the passage. We use the target answer by replacing it in the passage.

-Answer separation: Without replacing the target answer in the passage. We use the target answer by concatenating it at the end of the passage.

3.3 Evaluation

Following other QG studies, we use three evaluation metrics: BLEU, METEOR, and ROUGE$_L$ similar to other QG studies (Du et al., 2017; Kim et al., 2018). In the SQuAD dataset, a passage sometimes has multiple gold questions. Other studies used all these questions as references. In this research, we use one gold question that is a pair of the source passage and the target answer, as a reference. The score apparently decreases using this evaluation method. This method is more accurate to evaluate generated questions using target answers as inputs.

In addition, the evaluation method has one limitation. As we use interrogative phrases that form part of the questions, the score increases considerably because the model knows parts of the questions (interrogative phrases) to be generated. For this reason, we also evaluate the generated questions excluding the interrogative phrases. This treatment helps us verify whether the whole generated questions will lead to the target answers.

3.4 Result of the experiment

Table 1 shows the main result. Seq2Seq + Answer + Interrogative outperforms other methods by a large margin. Even when we evaluate questions excluding interrogative phrases, Seq2Seq + Answer + Interrogative still outperforms Seq2Seq + Answer. This result demonstrates that interrogative phrases help to improve the performance of QG. Furthermore, it is noteworthy that our simple Seq2Seq model outperforms the state-of-the-art model ASs2s. The results of the ablation test show that our components help to increase the accuracy. Notably, the copy mechanism increases the accuracy to a considerable extent, showing that it is very useful for the QG task. With regard to the answers, replacing target answers contributes

			Normal Evaluation			Excluding Interrogative Phrases		
Model	Answer	Interrogative	BLEU	METEOR	ROUGE$_L$	BLEU	METEOR	ROUGE$_L$
ASs2s	✓		14.6	19.1	43.0	13.7	17.7	40.8
	✓	✓	21.6	23.0	51.2	14.4	17.8	41.8
Seq2Seq			12.1	16.4	39.0	11.8	15.5	37.9
		✓	22.8	23.1	51.6	14.4	17.4	41.6
	✓		18.7	21.8	47.5	17.9	20.4	45.2
	✓	✓	**26.4**	**26.2**	**55.9**	**18.6**	**20.9**	**46.9**
Ablation Tests								
- Copy	✓	✓	22.0	23.3	52.1	14.2	17.9	42.8
- Coverage	✓	✓	26.4	26.1	56.0	18.5	20.8	46.9
- Answer	✓	✓	26.1	25.9	55.5	18.3	20.5	46.3
- Answer Sep	✓	✓	23.5	23.5	52.2	15.2	17.9	42.4

Table 1: Result of the QG. We use three evaluation metrics: BLEU, METEOR, and ROUGE$_L$. We also evaluate questions excluding interrogative phrases. In addition, we do four ablation tests.

	Seq2Seq+Answer	Seq2Seq+Answer +Interrogative
Interrogative Phrase		
Same as Gold	51%	100%
Appropriate	43%	0%
Inappropriate	6%	0%
Answer		
Correct	61%	74%
Incorrect	39%	26%

Table 2: Human evaluations of QG. We check the questions according to two criteria: (1) Do the interrogative phrases match the target answers? (2) Can the questions provide the target answers?

to increasing the accuracy compared to concatenating the target answers at the end of the passage.

Human evaluation We conduct human evaluation to check whether the generated questions are appropriate. We randomly extract 100 generated questions from the test set. We check the questions according to two criteria: (1) Do the interrogative phrases match the target answers? We classify the generated questions as follows: the same interrogative phrase as one of the gold questions, different from one of the gold questions but an appropriate interrogative phrase that matches the target answer, and inappropriate interrogative phrase that do not match the target answer. (2) Can the questions provide the target answers?

Table 2 shows the result. Seq2Seq + Answer + Interrogative always generates questions that include the same interrogative phrase as one of the gold questions. Seq2Seq2 + Answer often uses simpler interrogative phrases than Seq2Seq + Answer + Interrogative, and thus, the questions become abstractive. For example, while the gold interrogative phrase is "what year", Seq2Seq + Answer generates "when". With regard to the answers, Seq2Seq + Answer + Interrogative outperforms Seq2Seq + Answer. This implies that using interrogative phrases as inputs makes the questions more informative and answerable.

Passage: A regulation of the Rhine was called for, with an upper canal near Diepoldsau and a lower canal at Fuach, in order to counteract the constant flooding and strong sedimentation in the western Rhine Delta. **Target Answer:** constant flooding **Gold Question:** Why was the Rhine regulated? **ASs2s:** What is a lower canal by a lower canal at the western Rhine Delta? **Seq2Seq+Answer:** What was the regulation of the Rhine called for ? **Seq2Seq+Answer+Interrogative:** Why was a regulation of the Rhine called ?
Passage: [note 6] In 1978 a disco version of the theme was released in the UK, Denmark and Australia by the group Mankind, which reached number 24 in the UK charts. **Target Answer:** number 24 **Gold Question:** How high on the charts did the Mankind version of the theme go? **ASs2s:** What was the group of the theme ballad released by the group? **Seq2Seq+Answer:** How many number of the UK charts did the group Mankind reach in the UK charts? **Seq2Seq+Answer+Interrogative:** How high on the charts did the group Mankind reach?

Table 3: Examples of the generated questions. This shows questions generated by three models: ASs2S, Seq2Seq + Answer, and Seq2Seq + Answer + Interrogative.

Case study Table 3 shows two examples generated by our model. In the first example, ASs2s and Seq2Seq + Answer use an inappropriate interrogative word "what", and passages become unnatural. This is probably because the target answer is a short noun phrase, so the system decided to use it without considering the context. However, Seq2Seq + Answer + Interrogative uses the cor-

Passage: In July 1977, General Zia-ul-Haq overthrew Prime Minister Zulfiqar Ali Bhutto 's regime in Pakistan .
Interrogative Phrase: What **Generated Question:** What was the name of the Prime Minister who overthrew Pakistan in 1977?
Interrogative Phrase: Who **Generated Question:** Who overthrew Prime Minister in 1977?
Interrogative Phrase: When **Generated Question:** When was Zulfiqar Ali Bhutto elected?
Interrogative Phrase: Where **Generated Question:** Where was Zulfiqar Ali Bhutto from?
Interrogative Phrase: How **Generated Question:** How did Zulfiqar Ali Bhutto respond to the Prime Minister?
Interrogative Phrase: Why **Generated Question:** Why was Zulfiqar Ali Bhutto elected?
Interrogative Phrase: In what year **Generated Question:** In what year was Zulfiqar Ali Bhutto elected?
Interrogative Phrase: How many **Generated Question:** How many Prime Minister did Zulfiqar Ali Bhutto have in pakistan?

Table 4: Example of the question generated by the Seq2Seq + Interrogative model. This model does not use an answer phrase. Thus, it can generate various questions only from passages and interrogative phrases.

rect interrogative phrase and the generated question can thus provide the target answer.

In the second example, a similar phenomenon appears. ASs2s use "what" and Seq2Seq + Answer use "how many number of the UK charts" as interrogative phrases for the target answer "number 24". They are inappropriate for context, and the passage become unnatural.

Table 4 shows an example generated by Seq2Seq + Interrogative model. This example comprises a passage and eight questions generated from eight frequently interrogative phrases. This model does not use answer phrases. Thus, it can generate various questions only from passages and interrogative phrases. They can also generate questions where the answer phrase is not in the passage although all questions in the train data include answer phrases in the passage.

3.5 Question answering using generated questions

We also examine the quality of the generated questions in the question answering experiments. We verify whether the generated questions can provide the target answers. We apply BERT (Devlin et al., 2019) for the question answering model on the SQuAD dataset. We train and test the question answering model using the generated questions. We also train and test original passages created by a human in the SQuAD dataset as the upper bound. We use two evaluation metrics: Exact Match and F1 score.

Method	Answer	Interrogative	EM	F1
ASs2s	✓		68.9	78.7
	✓	✓	71.7	80.1
Seq2Seq			34.1	43.4
		✓	58.7	69.0
	✓		72.6	82.2
	✓	✓	**75.3**	**84.2**
Human			81.2	88.9

Table 5: Result of question answering using generated questions. We use two evaluation metrics: Exact Match and F1 score.

Table 5 shows the result. This demonstrates that using interrogative phrases helps the question answering system to reach the target answers. The score of Seq2Seq + Answer + Interrogative is a little lower than the one associated with the human-generated questions (by 4.7 F1 points). This shows that for the question answering model, the system-generated questions are close to those created by a human.

4 Conclusion and Future Work

In this study, we explore the use of interrogative phrases as additional sources to control QG. The results of the experiment show that using interrogative phrases contributes to improving the performance of QG. The question answering experiment demonstrates that using interrogative phrases helps the question answering system to provide target answers.

Future studies on QG will focus on the following aspects. (1) We directly input the interrogative phrases in this study. However, we also consider automatic selection of appropriate interrogative phrases such that the answerers can reach the target answer easily. (2) The model sometimes generates questions that contradict the source passages. To reduce that risk, we will use textual entailment to verify whether the generated questions are consistent with the source passages.

Acknowledgments

We thank the laboratory members for their useful comments. This work was supported by JSPS KAKENHI Grant Number 19H01118.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*.

Yllias Chali and Tina Baghaee. 2018. Automatic opinion question generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 152–158.

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*, pages 4171–4186.

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*, pages 1342–1352.

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.

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.

Yanghoon Kim, Hwanhee Lee, Joongbo Shin, and Kyomin Jung. 2018. Improving Neural Question Generation using Answer Separation. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of the 55th annual meeting of the Association for Computational Linguistics*, pages 67–72.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of the the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 55–60.

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

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1073–1083.

Linfeng Song, Zhiguo Wang, Wael Hamza, Yue Zhang, and Daniel Gildea. 2018. Leveraging Context Information for Natural Question Generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 569–574.

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.

Duyu Tang, Nan Duan, Zhao Yan, Zhirui Zhang, Yibo Sun, Shujie Liu, Yuanhua Lv, and Ming Zhou. 2018. Learning to collaborate for question answering and asking. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1564–1574.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 76–85.

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.