Let Me Know What to Ask: Interrogative-Word-Aware Question Generation

Junmo Kang[*] **Haritz Puerto San Roman**[*] **Sung-Hyon Myaeng**
School of Computing, KAIST
Daejeon, Republic of Korea
{junmo.kang, haritzpuerto94, myaeng}@kaist.ac.kr

Abstract

Question Generation (QG) is a Natural Language Processing (NLP) task that aids advances in Question Answering (QA) and conversational assistants. Existing models focus on generating a question based on a text and possibly the answer to the generated question. They need to determine the type of interrogative word to be generated while having to pay attention to the grammar and vocabulary of the question. In this work, we propose Interrogative-Word-Aware Question Generation (IWAQG), a pipelined system composed of two modules: an interrogative word classifier and a QG model. The first module predicts the interrogative word that is provided to the second module to create the question. Owing to an increased recall of deciding the interrogative words to be used for the generated questions, the proposed model achieves new state-of-the-art results on the task of QG in SQuAD, improving from 46.58 to 47.69 in BLEU-1, 17.55 to 18.53 in BLEU-4, 21.24 to 22.33 in METEOR, and from 44.53 to 46.94 in ROUGE-L.

1 Introduction

Question Generation (QG) is the task of creating questions about a text in natural language. This is an important task for Question Answering (QA) since it can help create QA datasets. It is also useful for conversational systems like Amazon Alexa. Due to the surge of interests in these systems, QG is also drawing the attention of the research community. One of the reasons for the fast advances in QA capabilities is the creation of large datasets like SQuAD (Rajpurkar et al., 2016) and TriviaQA (Joshi et al., 2017). Since the creation of such datasets is either costly if done manually or prone to error if done automatically, reliable and meaningful QG can play a key role in the advances of QA (Lewis et al., 2019).

[*]Equal contribution.

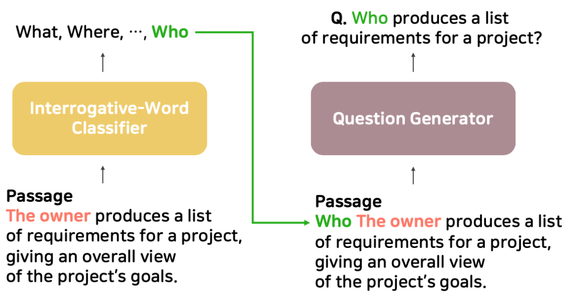
What, Where, ···, Who
Interrogative-Word Classifier
Passage
The owner produces a list of requirements for a project, giving an overall view of the project's goals.
Q. Who produces a list of requirements for a project?
Question Generator
Passage
Who The owner produces a list of requirements for a project, giving an overall view of the project's goals.

Figure 1: High-level overview of the proposed model.

QG is a difficult task due to the need for understanding of the text to ask about and generating a question that is grammatically correct and semantically adequate according to the given text. This task is considered to have two parts: *what to ask* and *how to ask*. The first one refers to the identification of relevant portions of the text to ask about. This requires machine reading comprehension since the system has to understand the text. The latter refers to the creation of a natural language question that is grammatically correct and semantically precise. Most of the current approaches utilize sequence-to-sequence models, composed of an encoder model that first transforms a passage into a vector and a decoder model that given this vector, generates a question about the passage (Liu et al., 2019; Sun et al., 2018; Zhao et al., 2018; Pan et al., 2019).

There are different settings for QG. Some authors like (Subramanian et al., 2018) assumes that only a passage is given, attempts to find candidate key phrases that represent the core of the questions to be created. Others follow an answer-aware setting, where the input is a passage and the answer to the question to create (Zhao et al., 2018). We assume this setting and consider that the answer is a span of the passage, as in SQuAD. Following this approach, the decoder of the sequence-to-sequence model has to learn to generate both the interrogative word (i.e., wh-word) and the rest of the question simultaneously.

The main claim of our work is that separating the two tasks (i.e., interrogative-word classification and question generation) can lead to a better performance. We posit that the interrogative word must be predicted by a well-trained classifier. We consider that selecting the right interrogative word is the key to generate high-quality questions. For example, a question with a wrong interrogative word for the answer "the owner" is: "what produces a list of requirements for a project?". However, with the right interrogative word, *who*, the question would be: "who produces a list of requirements for a project?", which is clear that is more adequate regarding the answer than the first one. According to our claim, the independent classification model can improve the recall of interrogative words of a QG model because 1) the interrogative word classification task is easier to solve than generating the interrogative word along with the full question in the QG model and 2) the QG model would be able to generate the interrogative word easily by using the copy mechanism, which can copy parts of the input of the encoder. With these hypotheses, we propose Interrogative-Word-Aware Question Generation (IWAQG), a pipelined system composed of two modules: an interrogative-word classifier that predicts the interrogative word and a QG model that generates a question conditioned on the predicted interrogative word. Figure 1 shows a high-level overview of our approach.

The proposed model achieves new state-of-the-art results on the task of QG in SQuAD, improving from 46.58 to 47.69 in BLEU-1, 17.55 to 18.53 in BLEU-4, 21.24 to 22.33 in METEOR, and from 44.53 to 46.94 in ROUGE-L.

2 Related Work

Question Generation (QG) problem has been approached in two ways. One is based on heuristics, templates and syntactic rules (Heilman and Smith, 2010; Mazidi and Nielsen, 2014; Labutov et al., 2015). This type of approach requires a heavy human effort, so they do not scale well. The other approach is based on neural networks and it is becoming popular due to the recent progress of deep learning in NLP (Pan et al., 2019). Du et al. (2017) is the first one to propose an sequence-to-sequence model to tackle the QG problem and outperformed the previous state-of-the-art model using human and automatic evaluations.

Sun et al. (2018) proposed a similar approach to us, an answer-aware sequence-to-sequence model with a special decoding mode in charge of only the interrogative word. However, we propose to predict the interrogative word before the encoding stage, so that the decoder can focus more on the rest of the question rather than on the interrogative word. Besides, they cannot train the interrogative-word classifier using golden labels because it is learned implicitly inside the decoder. Duan et al. (2017) proposed, in a similar way to us, a pipeline approach. First, the authors create a long list of question templates like "who is author of", and "who is wife of". Then, when generating the question, they select first the question template and next, they fill it in. To select the question template, they proposed two approaches. One is a retrieval-based question pattern prediction, and the second one is a generation-based question pattern prediction. The first one has the problem that is computationally expensive when the question pattern size is large, and the second one, although it yields to better results, it is a generative approach and we argue that just modeling the interrogative word prediction as a classification task is easier and can lead to better results. As far as we know, we are the first one to propose an explicit interrogative-word classifier that provides the interrogative word to the question generator.

3 Interrogative-Word-Aware Question Generation

3.1 Problem Statement

Given a passage P, and an answer A, we want to find a question Q, whose answer is A. More formally:

$$\overline{Q} = \arg\max_{Q} Prob(Q|P, A)$$

We assume that P is a paragraph composed of a list of words: $P = \{x_t\}_{t=1}^{M}$, and the answer is a subspan of P.

We model this problem with a pipelined approach. First, given P and A, we predict the interrogative word I_w, and then, we input into QG module P, A, and I_w. The overall architecture of our model is shown in 2.

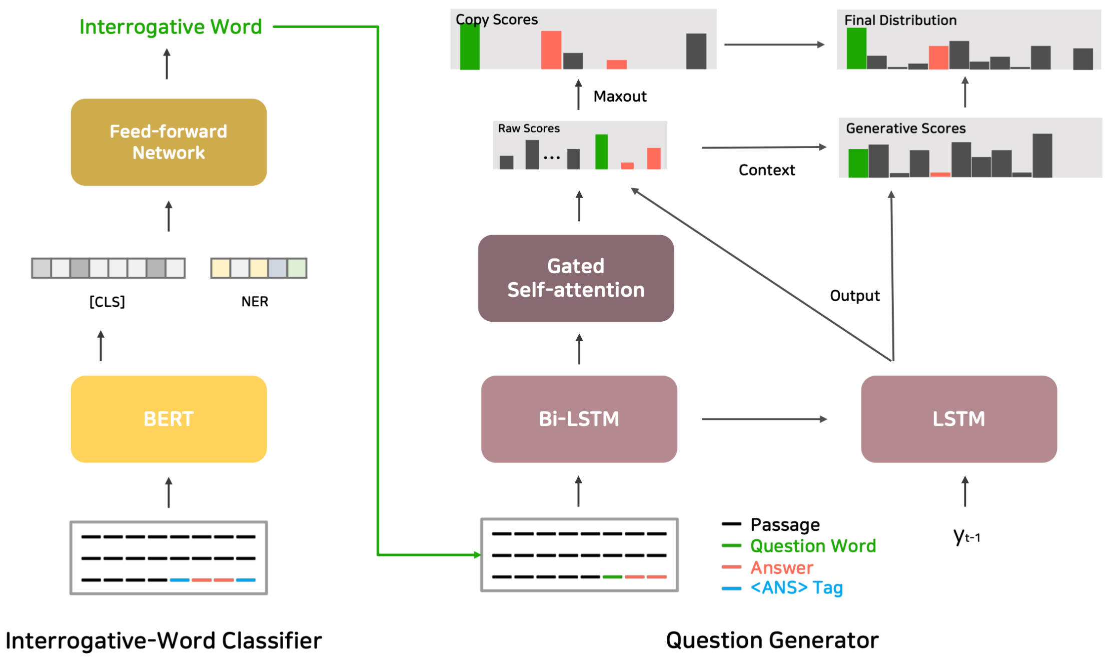

Figure 2: Overall architecture of IWAQG.

3.2 Interrogative-Word Classifier

As discussed in section 5.2, any model can be used to predict interrogative words if its accuracy is high enough. Our interrogative-word classifier is based on BERT, a state-of-the-art model in many NLP tasks that can successfully utilize the context to grasp the semantics of the words inside a sentence (Devlin et al., 2018). We input a passage that contains the answer of the question we want to build and add the special token `[ANS]` to let BERT knows that the answer span has a special meaning and must be used differently to the rest of the passage. As required by BERT, the first token of the input is the special token `[CLS]`, and the last is `[SEP]`. This `[CLS]` token embedding originally was designed for classification tasks. In our case, to classify interrogative words, it learns how to represent the context and the answer information.

On top of BERT, we build a feed-forward network that receives as input the `[CLS]` token embedding concatenated with a learnable embedding of the entity type of the answer, as shown on the left side of Figure 2. We propose to utilize the entity type of the answer because there is a clear correlation between the answer type of the question and the entity type of the answer. For example, if the interrogative word is *who*, the answer is very likely to have an entity type *person*. Since we are using `[CLS]` token embedding as a representation of the context and the answer, we consider that using an explicit entity type embedding of the answer could help the system.

3.3 Question Generator

For the QG module, we employ one of the current state-of-the-art QG models (Zhao et al., 2018). This model is a sequence-to-sequence neural network that uses a gated self-attention in the encoder and an attention mechanism with maxout pointer in the decoder.

One way to connect the interrogative-word classifier to the QG model is to use the predicted interrogative word as the first output token of the decoder by default. However, we cannot expect a perfect interrogative-word classifier and also, the first word of the questions is not necessarily an interrogative word. Therefore, in this work, we add the predicted interrogative word to the input of the QG model to let the model decide whether to use it or not. In this way, we can condition the generated question on the predicted interrogative word effectively.

3.3.1 Encoder

The encoder is composed of a Recurrent Neural Network (RNN), a self-attention network, and a feature fusion gate (Gong and Bowman, 2018). The goal of this fusion gate is to combine two intermediate learnable features into the final encoded passage-answer representation. The input of this model is the passage P. It includes the answer and the predicted interrogative word I_w, which is located just before the answer span. The RNN receives the word embedding of the tokens of this text concatenated with a learnable meta-embedding that tags if the token is the interrogative word, the answer of the question to generate or the context of the answer.

3.3.2 Decoder

The decoder is composed of an RNN with an attention layer and a copy mechanism (Gu et al., 2016). The RNN of the decoder at time step t receives its hidden state at the previous time step $t-1$ and the previously generated output y_{t-1}. At $t = 0$, it receives the last hidden state of the encoder. This model combines the probability of generating a word and the probability of copying that word from the input as shown on the right side of Figure 2. To compute the generative scores, it uses the outputs of the decoder, and the context of the encoder, which is based on the raw attention scores. To compute the copy scores, it uses the outputs of the RNN and the raw attention scores of the encoder. Zhao et al. (2018) observed that the repetition of words in the input sequence tends to create repetitions in the output sequence too. Thus, they proposed a maxout pointer mechanism instead of the regular pointer mechanism (Vinyals et al., 2015). This new pointer mechanism limits the magnitude of the scores of the repeated words to their maximum value. To do that, first, the attention scores are computed over the input sequence and then, the score of a word at time step t is calculated as the maximum of all scores pointing to the same word in the input sequence. The final probability distribution is calculated by applying the softmax function on the concatenation of copy scores and generative scores and summing up the probabilities pointing to the same words.

4 Experiments

In our experiments, we study our proposed system on SQuAD dataset v1.1. (Rajpurkar et al., 2016), prove the validity of our hypothesis and compare it with the current state of the art.

4.1 Dataset

In order to train our interrogative-word classifier, we use the training set of SQuAD v1.1 (Rajpurkar et al., 2016). This dataset is composed of 87599 instances, however, the number of interrogative words is not balanced as seen in 1. To train the interrogative-word classifier, we downsample the training set to have a balanced dataset.

Class	Original	After Downsampling
What	50385	4000
Which	6111	4000
Where	3731	3731
When	5437	4000
Who	9162	4000
Why	1224	1224
How	9408	4000
Others	9408	4000

Table 1: SQuAD training set statistics. Full training set and downsampled training set.

For a fair comparison with previous models, we train the QG model on the training set of SQuAD and split by half the dev set into dev and test randomly as Zhou et al. (2017).

4.2 Implementation

The interrogative-word classifier is made using the PyTorch implementation of BERT-base-uncased made by HuggingFace[1]. It was trained for three epochs using cross entropy loss as the objective function. The entity types are obtained using spaCy[2]. If spaCy cannot return an entity for a given answer, we label it as `None`. The dimension of the entity type embedding is 5. The input dimension of the classifier is 773 (768 from BERT base hidden size and 5 from the entity type embedding size) and the output dimension is 8 since we predict the interrogative words: *what*, *which*, *where*, *when*, *who*, *why*, *how*, and *others*. The feed-forward network consists of a single layer. For optimization, we used Adam optimizer with weight decay and learning rate of 5e-5. The QG model is based on the model proposed by (Zhao et al., 2018) with small modifications using PyTorch. The encoder uses a BiLSTM and the decoder uses an LSTM. During training, the QG model uses the golden interrogative words to enforce the decoder to always copy the interrogative word. On the other hand, during inference, it uses

[1]https://github.com/huggingface/pytorch-transformers
[2]https://spacy.io/

the interrogative word predictions from the classifier.

4.3 Evaluation

We perform an automatic evaluation using the metrics: BLUE-1, BLUE-2, BLUE-3, BLUE-4 (Papineni et al., 2002), METEOR (Lavie and Denkowski, 2009) and ROUGE-L (Lin, 2004). In addition, we perform a qualitative analysis where we compare the generated questions of the baseline (Zhao et al., 2018), our proposed model, the upper bound performance of our model, and the golden question.

5 Results

5.1 Comparison with Previous Models

Our interrogative-word classifier achieves an accuracy of 73.8% on the test set of SQuAD. Using this model for the pipelined system, we compare the performance of the QG model with respect to the previous state-of-the-art models. Table 2 shows the evaluation results of our model and the current state-of-the-art models, which are briefly described below.

- Zhou et al. (2017) is one of the first authors who proposed a sequence-to-sequence model with attention and copy mechanism. They also proposed the use of POS and NER tags as lexical features for the encoder.
- Zhao et al. (2018) proposed the model in which we based our QG module.
- Kim et al. (2019) proposed QG architecture that treats the passage and the target answer separately.
- Liu et al. (2019) proposed a sequence-to-sequence model with a clue word predictor using a Graph Convolutional Networks to identify if each word in the input passage is a potential clue that should be copied into the generated question.

Our model outperforms all other models in all the metrics. This improvement is consistent, around 2%. This is due to the improvement in the recall of the interrogative words. All these measures are based on the overlap between the golden question and the generated question, so using the right interrogative word, we can improve these scores. In addition, generating the right interrogative word also helps to create better questions since the output of the RNN of the decoder at time step t also depends on the previously generated word.

5.2 Upper Bound Performance of IWAQG

We analyze the upper bound improvement that our QG model can have according to different levels of accuracy of the interrogative-word classifier. In order to do that, instead of using our interrogative-word classifier, we use the golden labels of the test set and generated noise to simulate a classifier with different accuracy levels. Table 3 and Figure 3 show a linear relationship between the accuracy of the classifier and the IWAQG. This demonstrates the effectiveness of our pipelined approach regardless of the interrogative-word classifier model.

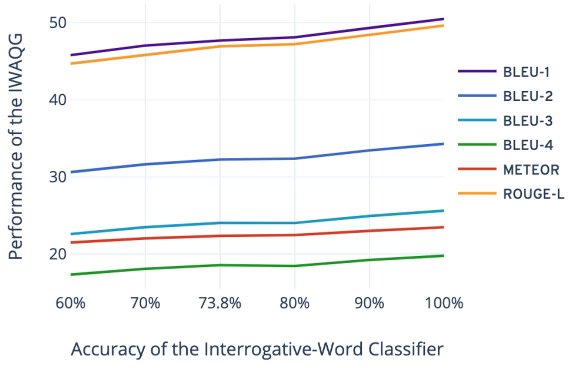

Figure 3: Performance of the QG model with respect to the accuracy of the interrogative-word classifier.

In addition, we analyze the recall of the interrogative words generated by our pipelined system. As shown in the Table 4, the total recall of using only the QG module is 68.29%, while the recall of our proposed system, IWAQG, is 74.10%, an improvement of almost 6%. Furthermore, if we assume a perfect interrogative-word classifier, the recall would be 99.72%, a dramatic improvement which proves the validity of our hypothesis.

5.3 Effectiveness of the input of interrogative words into the QG model

In this section, we show the effectiveness of inserting explicitly the predicted interrogative word into the passage. We argue that this simple way of connecting the two models exploits the characteristics of the copy mechanism successfully. As we can

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE-L
Zhou et al. (2017)	-	-	-	13.29	-	-
Zhao et al. (2018)*	45.69	29.58	22.16	16.85	20.62	44.99
Kim et al. (2019)	-	-	-	16.17	-	-
Liu et al. (2019)	46.58	30.90	22.82	17.55	21.24	44.53
IWAQG	**47.69**	**32.24**	**24.01**	**18.53**	**22.33**	**46.94**

Table 2: Comparison of our model with the baselines. "*" is our QG module.

Accuracy	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE-L
Only QG*	**45.63**	**30.43**	**22.51**	**17.30**	**21.06**	**45.42**
60%	45.80	30.61	22.57	17.30	21.47	44.70
70%	47.05	31.62	23.46	18.05	22.00	45.88
IWAQG (73.8%)	**47.69**	**32.24**	**24.01**	**18.53**	**22.33**	**46.94**
80%	48.11	32.36	24.00	18.42	22.43	47.22
90%	49.33	33.43	24.91	19.20	22.98	48.41
Upper Bound (100%)	**50.51**	**34.28**	**25.60**	**19.75**	**23.45**	**49.65**

Table 3: Performance of the QG model with respect to the accuracy of the interrogative-word classifier. "*" is our implementation of the QG module without our interrogative-word classifier (Zhao et al., 2018).

see in Figure 4, the attention score of the generated interrogative word, *who*, is relatively high for the predicted interrogative word and lower for the other words. This means that it is very likely that the interrogative word inserted into the passage is copied as intended.

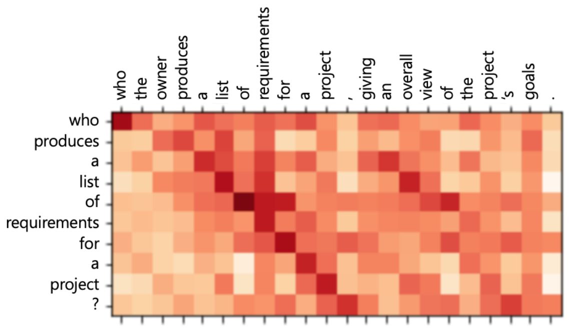

Figure 4: Attention matrix between the generated question (Y-axis) and the given passage (X-axis).

5.4 Qualitative Analysis

In this section, we present a sample of the generated questions of our model, the upper bound model (interrogative-word classifier accuracy is 100%), the baseline (Zhao et al., 2018), and the golden questions to show how our model improves the recall of the interrogative words with respect to the baseline. In general, our model has a better recall of interrogative words than the baseline which leads us to a better quality of questions. However, since we are still far from a perfect interrogative-word classifier, we also show that questions that our current model cannot generate correctly could be generated well if we had a better classifier.

As we can see in Table 5, in the first three examples the interrogative words generated by the baseline are wrong, while our model is right. In addition, due to the wrong selection of interrogative words, in the second example, the topic of the question generated by the baseline is also wrong. On the other hand, since our model selects the right interrogative word, it can create the right question. Each generated word depends on the previously generated word because of the generative LSTM model, so it is very important to select correctly the first word, i.e. the interrogative word. However, the performance of our proposed interrogative-word classifier is not perfect, if it had a 100% accuracy, then, we could improve the quality of the generated questions like in the last two examples.

5.5 Ablation Study

We tried to combine different features shown in Table 6 for the interrogative-word classifier. In this section, we analyze their impact on the performance of the model.

The first model is only using the `[CLS]` BERT token embedding (Devlin et al., 2018) that represents the input passage. In this model, the input

Model	What	Which	Where	When	Who	Why	How	Others	Total
Only QG*	82.24%	0.29%	51.90%	60.82%	68.34%	12.66%	60.62%	2.13%	68.29%
IWAQG	87.66%	1.46%	66.24%	49.41%	76.41%	50.63%	70.26%	14.89%	74.10%
Upper Bound	99.87%	99.71%	100.00%	99.71%	99.84%	98.73%	99.67%	89.36%	99.72%

Table 4: Recall of interrogative words of the QG model. "*" is our implementation of the QG module without our interrogative-word classifier (Zhao et al., 2018).

is the passage where the answer appears but, the model does not know where the answer is. The second model is the previous one with the entity type of the answer as an additional feature. The performance of this model is a bit better than the first one but it is not enough to be utilized effectively for our pipeline. In the third model, the input is the passage. This model uses the average of the answer token embeddings generated by BERT along with the `[CLS]` token embedding. As we can see, the performance noticeably increased, which indicates that answer information is the key to predict the interrogative word needed. In the fourth model, we added the special token `[ANS]` at the beginning and at the end of the answer span to let BERT knows where the answer is in the passage. So the input to the feed-forward network is only the `[CLS]` token embedding. This model clearly outperforms the previous one, which shows that BERT can exploit the answer information better if it is tagged with the `[ANS]` token. The fifth model is the same as the previous one but with the addition of the entity-type embedding of the answer. The combination of the three features (answer, answer entity type, and passage) yields to the best performance.

Classifier	Accuracy
CLS	56.0%
CLS + NER	56.6%
CLS + AE	70.3%
CLS + AT	73.3%
CLS + AT + NER	**73.8%**

Table 6: Ablation Study of our interrogative-word classifier.

In addition, we provide the recall and precision per class for our final interrogative-word classifier (CLS + AT in Table 7). As we can see, the overall recall is high, and it is also higher than just using the QG module (Table 4), which proves our hypothesis that modeling the interrogative-word prediction task as an independent classification problem yields to a higher recall than generating them with the full question. However, the recall of *which* is very low. This is due to the intrinsic difficulty of predicting this interrogative words. Questions like "what country" and "which country" can be correct depending on the context, but the meaning is very similar. Our model has also problem with *why* due to the lack of training instances for this class. Lastly, the recall of '*when* is also low because many questions of this type can be formulated with other interrogative words, e.g.: instead of "When did WWII start?", we can ask "In which year did WWII start?".

Class	Recall	Precision
What	87.7%	76.0%
Which	1.4%	38.0%
Where	65.9%	55.8%
When	49.2%	69.8%
Who	76.9%	66.7%
Why	50.1%	74.1%
How	70.5%	79.0%
Others	10.5%	57.0%

Table 7: Recall and precision of interrogative words of our interrogative-word classifier.

6 Conclusion and Future Work

In this work, we proposed an Interrogative-Word-Aware Question Generation (IWAQG), a pipelined model composed of an interrogative-word classifier and a question generator to tackle the question generation task. First, we predict the interrogative word. Then, the Question Generation (QG) model generates the question using the predicted interrogative word. Thanks to this independent interrogative-word classifier and the copy mechanism of the question generation model, we are able to improve the recall of the interrogative words in the generated questions. This improvement also leads to a better quality of the generated questions. We prove our hypotheses through quantitative and qualitative experiments, showing that our pipelined system outperforms the previous state-of-the-art models. Lastly, we also prove that

id	Only QG*	IWAQG	Upper Bound	Golden	Answer
1	what produces a list of requirements for a project?	who produces a list of requirements for a project?	who produces a list of requirements for a project?	who produces a list of requirements for a project, giving an overall view of the project's goals?	The owner
2	how many tunnels were constructed through newcastle city centre?	what type of tunnels constructed through newcastle city centre?	what type of tunnels constructed through newcastle city centre ?	what type of tunnels are constructed through newcastle 's city center?	deep-level tunnels
3	who received a battering during the siege of newcastle?	what received a battering during the siege of newcastle ?	what received a battering during the siege of newcastle ?	what received a battering during the siege of newcastle?	The church tower
4	what system is newcastle international airport connected to?	what system is newcastle international airport connected to?	how is newcastle international airport connected to ?	how is newport 's airport connected to the city?	via the Metro Light Rail system
5	who was the country most dependent on arab oil?	what country was the most dependent on arab oil?	which country was the most dependent on arab oil?	which country is the most dependent on arab oil?	Japan

Table 5: Qualitative Analysis. Comparison between the baseline, our proposed model, the upper bound of our model, the golden question and the answer of the question. "*" is our implementation of the QG module without our interrogative-word classifier (Zhao et al., 2018).

our methodology is remarkably effective, showing a theoretical upper bound of the potential improvement using a more accurate interrogative-word classifier.

In the future, we would like to improve the interrogative-word classifier, since it would clearly improve the performance of the whole system as we showed. We also expect that the use of the Transformer architecture(Vaswani et al., 2017) could improve the QG model. In addition, we plan to test our approach on other datasets to prove its generalization capability. Finally, an interesting application of this work could be to utilize QG to improve Question Answering systems.

Acknowledgements

This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (2017M3C4A7065962).

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Association for Computational Linguistics (ACL)*.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. In *EMNLP*.

Yichen Gong and Samuel Bowman. 2018. Ruminating reader: Reasoning with gated multi-hop attention. In *Proceedings of the Workshop on Machine*

Reading for Question Answering, pages 1–11, Melbourne, Australia. Association for Computational Linguistics.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

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.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

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.

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*, pages 889–898.

Alon Lavie and Michael J. Denkowski. 2009. The meteor metric for automatic evaluation of machine translation. *Machine Translation*, 23(2-3):105–115.

Patrick Lewis, Ludovic Denoyer, and Sebastian Riedel. 2019. Unsupervised question answering by cloze translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Bang Liu, Mingjun Zhao, Di Niu, Kunfeng Lai, Yancheng He, Haojie Wei, and Yu Xu. 2019. Learning to generate questions by learningwhat not to generate. In *The World Wide Web Conference*, WWW '19, pages 1106–1118, New York, NY, USA. ACM.

Karen Mazidi and Rodney Nielsen. 2014. Linguistic considerations in automatic question generation. In *52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference*, volume 2.

Liangming Pan, Wenqiang Lei, Tat-Seng Chua, and Min-Yen Kan. 2019. Recent advances in neural question generation. *arXiv preprint arXiv:1905.08949*.

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

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

Sandeep Subramanian, Tong Wang, Xingdi Yuan, Saizheng Zhang, Adam Trischler, and Yoshua Bengio. 2018. Neural models for key phrase extraction and question generation. In *Proceedings of the Workshop on Machine Reading for Question Answering*, pages 78–88.

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 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.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

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 *NLPCC*.