Neural Question Generation from Text: A Preliminary Study

Qingyu Zhou[1][*], Nan Yang[2], Furu Wei[2], Chuanqi Tan[3], Hangbo Bao[1], and Ming Zhou[2]

[1] Harbin Institute of Technology, Harbin, China
qyzhgm@gmail.com, baohangbo@hit.edu.cn
[2] Microsoft Research, Beijing, China
nanya, fuwei, mingzhou@microsoft.com
[3] Beihang University, Beijing, China
tanchuanqi@nlsde.buaa.edu.cn

Abstract. Automatic question generation aims to generate questions from a text passage where the generated questions can be answered by certain sub-spans of the given passage. Traditional methods mainly use rigid heuristic rules to transform a sentence into related questions. In this work, we propose to apply the neural encoder-decoder model to generate meaningful and diverse questions from natural language sentences. The encoder reads the input text and the answer position, to produce an answer-aware input representation, which is fed to the decoder to generate an answer focused question. We conduct a preliminary study on neural question generation from text with the SQuAD dataset, and the experiment results show that our method can produce fluent and diverse questions.

1 Introduction

Automatic question generation from natural language text aims to generate questions taking text as input, which has the potential value of education purpose [9]. As the reverse task of question answering, question generation also has the potential for providing a large scale corpus of question-answer pairs.

Previous works for question generation mainly use rigid heuristic rules to transform a sentence into related questions [9, 2]. However, these methods heavily rely on human-designed transformation and generation rules, which cannot be easily adopted to other domains. Instead of generating questions from texts, [19] proposed a neural network method to generate factoid questions from structured data.

In this work we conduct a preliminary study on question generation from text with neural networks, which is denoted as the Neural Question Generation (NQG) framework, to generate natural language questions from text without predefined rules. The Neural Question Generation framework extends the sequence-to-sequence models by enriching the encoder with answer and lexical features to

---

[*] Contribution during internship at Microsoft Research.

generate answer focused questions. Concretely, the encoder reads not only the input sentence, but also the answer position indicator and lexical features. The answer position feature denotes the answer span in the input sentence, which is essential to generate answer relevant questions. The lexical features include part-of-speech (POS) and named entity (NER) tags to help produce better sentence encoding. Lastly, the decoder with attention mechanism [1] generates an answer specific question of the sentence.

Large-scale manually annotated passage and question pairs play a crucial role in developing question generation systems. We propose to adapt the recently released Stanford Question Answering Dataset (SQuAD) [18] as the training and development datasets for the question generation task. In SQuAD, the answers are labeled as subsequences in the given sentences by crowed sourcing, and it contains more than 100K questions which makes it feasible to train our neural network models. We conduct the experiments on SQuAD, and the experiment results show the neural network models can produce fluent and diverse questions from text.

2 Approach

In this section, we introduce the NQG framework, which consists of a feature-rich encoder and an attention-based decoder. Figure 1 provides an overview of our NQG framework.
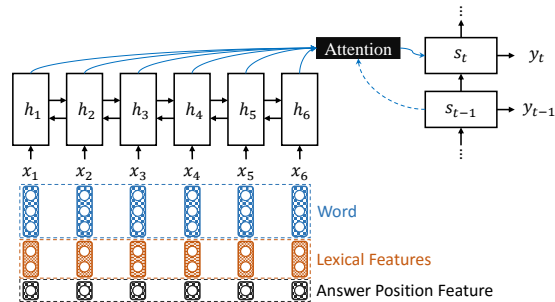

Fig. 1. Overview of the Neural Question Generation (NQG) framework.

2.1 Feature-Rich Encoder

In the NQG framework, we use Gated Recurrent Unit (GRU) [4] to build the encoder. To capture more context information, we use bidirectional GRU (Bi-GRU) to read the inputs in both forward and backward orders. Inspired by [3, 14], the BiGRU encoder not only reads the sentence words, but also handcrafted features, to produce a sequence of word-and-feature vectors. We concatenate

the word vector, lexical feature embedding vectors and answer position indicator embedding vector as the input of BiGRU encoder. Concretely, the BiGRU encoder reads the concatenated sentence word vector, lexical features, and answer position feature, $x = (x_1, x_2, \ldots, x_n)$, to produce two sequences of hidden vectors, i.e., the forward sequence $(\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_n)$ and the backward sequence $(\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_n)$. Lastly, the output sequence of the encoder is the concatenation of the two sequences, i.e., $h_i = [\boldsymbol{h}_i; \boldsymbol{h}_i]$.

Answer Position Feature To generate a question with respect to a specific answer in a sentence, we propose using answer position feature to locate the target answer. In this work, the BIO tagging scheme is used to label the position of a target answer. In this scheme, tag B denotes the start of an answer, tag I continues the answer and tag O marks words that do not form part of an answer. The BIO tags of answer position are embedded to real-valued vectors throu and fed to the feature-rich encoder. With the BIO tagging feature, the answer position is encoded to the hidden vectors and used to generate answer focused questions.

Lexical Features Besides the sentence words, we also feed other lexical features to the encoder. To encode more linguistic information, we select word case, POS and NER tags as the lexical features. As an intermediate layer of full parsing, POS tag feature is important in many NLP tasks, such as information extraction and dependency parsing [12]. Considering that SQuAD is constructed using Wikipedia articles, which contain lots of named entities, we add NER feature to help detecting them.

2.2 Attention-Based Decoder

We employ an attention-based GRU decoder to decode the sentence and answer information to generate questions. At decoding time step t, the GRU decoder reads the previous word embedding w_{t-1} and context vector c_{t-1} to compute the new hidden state s_t. We use a linear layer with the last backward encoder hidden state $\boldsymbol{h}_1$ to initialize the decoder GRU hidden state. The context vector c_t for current time step t is computed through the concatenate attention mechanism [11], which matches the current decoder state s_t with each encoder hidden state h_i to get an importance score. The importance scores are then normalized to get the current context vector by weighted sum:

$$s_t = \mathrm{GRU}(w_{t-1}, c_{t-1}, s_{t-1}) \tag{1}$$

$$s_0 = \tanh(\mathbf{W}_d \boldsymbol{h}_1 + b) \tag{2}$$

$$e_{t,i} = v_a^\top \tanh(\mathbf{W}_a s_{t-1} + \mathbf{U}_a h_i) \tag{3}$$

$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{i=1}^n \exp(e_{t,i})} \tag{4}$$

$$c_t = \sum_{i=1}^n \alpha_{t,i} h_i \tag{5}$$

We then combine the previous word embedding w_{t-1}, the current context vector c_t, and the decoder state s_t to get the readout state r_t. The readout state is passed through a maxout hidden layer [7] to predict the next word with a softmax layer over the decoder vocabulary:

$$r_t = \mathbf{W}_r w_{t-1} + \mathbf{U}_r c_t + \mathbf{V}_r s_t \tag{6}$$

$$m_t = [\max\{r_{t,2j-1}, r_{t,2j}\}]_{j=1,\ldots,d}^\top \tag{7}$$

$$p(y_t|y_1,\ldots,y_{t-1}) = \mathrm{softmax}(\mathbf{W}_o m_t) \tag{8}$$

where r_t is a $2d$-dimensional vector.

2.3 Copy Mechanism

To deal with the rare and unknown words problem, [8] propose using pointing mechanism to copy rare words from source sentence. We apply this pointing method in our NQG system. When decoding word t, the copy switch takes current decoder state s_t and context vector c_t as input and generates the probability p of copying a word from source sentence:

$$p = \sigma(\mathbf{W}s_t + \mathbf{U}c_t + b) \tag{9}$$

where σ is sigmoid function. We reuse the attention probability in equation 4 to decide which word to copy.

3   Experiments and Results

We use the SQuAD dataset as our training data. SQuAD is composed of more than 100K questions posed by crowd workers on 536 Wikipedia articles. We extract sentence-answer-question triples to build the training, development and test sets[4]. Since the test set is not publicly available, we randomly halve the development set to construct the new development and test sets. The extracted training, development and test sets contain 86,635, 8,965 and 8,964 triples respectively. We introduce the implementation details in the appendix.

We conduct several experiments and ablation tests as follows:

PCFG-Trans The rule-based system[4] modified on the code released by [9]. We modified the code so that it can generate question based on a given word span.

s2s+att We implement a seq2seq with attention as the baseline method.

NQG We extend the s2s+att with our feature-rich encoder to build the NQG system.

NQG+ Based on NQG, we incorporate copy mechanism to deal with rare words problem.

---

[4] We re-distribute the processed data split and PCFG-Trans baseline code at http://res.qyzhou.me

NQG+Pretrain Based on NQG+, we initialize the word embedding matrix with pre-trained GloVe [17] vectors.

NQG+STshare Based on NQG+, we make the encoder and decoder share the same embedding matrix.

NQG++ Based on NQG+, we use both pre-train word embedding and STshare methods, to further improve the performance.

NQG−Answer Ablation test, the answer position indicator is removed from NQG model.

NQG−POS Ablation test, the POS tag feature is removed from NQG model.

NQG−NER Ablation test, the NER feature is removed from NQG model.

NQG−Case Ablation test, the word case feature is removed from NQG model.

3.1 Implementation Details

Model Parameters We use the same vocabulary for both encoder and decoder. The vocabulary is collected from the training data and we keep the top 20,000 frequent words. We set the word embedding size to 300 and all GRU hidden state sizes to 512. The lexical and answer position features are embedded to 32-dimensional vectors. We use dropout [20] with probability $p = 0.5$. During testing, we use beam search with beam size 12.

Lexical Feature Annotation We use Stanford CoreNLP v3.7.0 [5] [13] to annotate POS and NER tags in sentences with its default configuration and pre-trained models.

Model Training We initialize model parameters randomly using a Gaussian distribution with Xavier scheme [6]. We use a combination of Adam [10] and simple SGD as our the optimizing algorithms. The training is separated into two phases, the first phase is optimizing the loss function with Adam and the second is with simple SGD. For the Adam optimizer, we set the learning rate $\alpha = 0.001$, two momentum parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$ respectively, and $\epsilon = 10^{-8}$. We use Adam optimizer until the BLEU score on the development set drops for six consecutive tests (we test the BLEU score on the development set for every 1,000 batches). Then we switch to a simple SGD optimizer with initial learning rate $\alpha = 0.5$ and halve it if the BLEU score on the development set drops for twelve consecutive tests. We also apply gradient clipping [16] with range $[-5, 5]$ for both Adam and SGD phases. To both speed up the training and converge quickly, we use mini-batch size 64 by grid search.

3.2 Human Evaluation

We evaluate the PCFG-Trans baseline and NQG++ with human judges. The rating scheme is provided in Table 1.

---

[5] https://stanfordnlp.github.io/CoreNLP/

Score	Rating scheme
3-Good	The question is meaningful and matches the sentence and answer very well
2-Borderline	The question matches the sentence and answer, more or less
1-Bad	The question either does not make sense or matches the sentence and answer

Table 1. Human rating scheme.

The human judges are asked to label the generated questions if they match the given sentence and answer span according to the rating scheme and examples. We provide some example questions with different scores in Table 2. For the first score 3 example, the question makes sense and the target answer "reason" can be used to answer it given the input sentence. For the second score 2 example, the question is inadequate for answering the sentence since the answer is about prime number. However, given the sentence, a reasonable person will give the targeted answer of the question. For the third score 1 example, the question is totally wrong given the sentence and answer.

Score	Sentence and generated question
3	**I:** -lsb- ... -rsb- for <u>reason</u> is the greatest enemy that faith has ; it never comes to the aid of spiritual things . " **O:** what is the biggest enemy that faith has have ?
2	**I:** in all other rows -lrb- a = 1 , 2 , 4 , 5 , 7 , and 8 -rrb- there are <u>infinitely many prime numbers</u> . **O:** how many numbers are in all other rows ?
1	**I:** while genghis khan never conquered all of china , his <u>grandson</u> kublai khan completed that conquest and established the yuan dynasty that is often credited with re-uniting china . **O:** who did kublai khan defeat that conquered all of china ?

Table 2. Human rating scheme examples, **I** is input, **O** is output, the underlined words are target answers.

3.3 Results and Analysis

We report BLEU-4 score [15] as the evaluation metric of our NQG system.

Table 3 shows the BLEU-4 scores of different settings. We report the beam search results on both development and test sets. Our NQG framework outperforms the PCFG-Trans and s2s+att baselines by a large margin. This shows that

the lexical features and answer position indicator can benefit the question generation. With the help of copy mechanism, NQG+ has a 2.05 BLEU improvement since it solves the rare words problem. The extended version, NQG++, has 1.11 BLEU score gain over NQG+, which shows that initializing with pre-trained word vectors and sharing them between encoder and decoder help learn better word representation.

Model	Dev set	Test set
PCFG-Trans	9.28	9.31
s2s+att	3.01	3.06
NQG	10.06	10.13
NQG+	12.30	12.18
NQG+Pretrain	12.80	12.69
NQG+STshare	12.92	12.80
NQG++	**13.27**	**13.29**
NQG−Answer	2.79	2.98
NQG−POS	9.83	9.87
NQG−NER	9.50	9.29
NQG−Case	9.91	9.89

Table 3. BLEU evaluation scores of baseline methods, different NQG framework configurations and some ablation tests.

Human Evaluation We evaluate the PCFG-Trans baseline and NQG++ with human judges. The rating scheme is, Good (3) - The question is meaningful and matches the sentence and answer very well; Borderline (2) - The question matches the sentence and answer, more or less; Bad (1) - The question either does not make sense or matches the sentence and answer. We provide more detailed rating examples in the supplementary material. Three human raters labeled 200 questions sampled from the test set to judge if the generated question matches the given sentence and answer span. The inter-rater aggreement is measured with Fleiss' kappa [5].

Table 4 reports the human judge results. The kappa scores show a moderate agreement between the human raters. Our NQG++ outperforms the PCFG-Trans baseline by 0.76 score, which shows that the questions generated by NQG++ are more related to the given sentence and answer span.

Ablation Test The answer position indicator, as expected, plays a crucial role in answer focused question generation as shown in the NQG−Answer ablation test. Without it, the performance drops terribly since the decoder has no information about the answer subsequence.

Ablation tests, NQG−Case, NQG−POS and NQG−NER, show that word case, POS and NER tag features contributes to question generation.

Model	AvgScore	Fleiss' kappa
PCFG-Trans	1.42	0.50
NQG++	2.18	0.46

Table 4. Human evaluation results.

Case Study Table 5 provides three examples generated by NQG++. The words with underline are the target answers. These three examples are with different question types, namely WHEN, WHAT and WHO respectively. It can be observed that the decoder can 'copy' spans from input sentences to generate the questions. Besides the underlined words , other meaningful spans can also be used as answer to generate correct answer focused questions.

I: in 1226 , immediately after returning from the west , genghis khan began a retaliatory attack on the tanguts .
G: in which year did genghis khan strike against the tanguts ?
O: in what year did genghis khan begin a retaliatory attack on the tanguts ?
I: in week 10 , manning suffered a partial tear of the plantar fasciitis in his left foot .
G: in the 10th week of the 2015 season , what injury was peyton manning dealing with ?
O: what did manning suffer in his left foot ?
I: like the lombardi trophy , the " 50 " will be designed by tiffany & co. .
G: who designed the vince lombardi trophy ?
O: who designed the lombardi trophy ?

Table 5. Examples of generated questions, I is the input sentence, G is the gold question and O is the NQG++ generated question. The underlined words are the target answers.

Type of Generated Questions Following [21], we classify the questions into different types, i.e., WHAT, HOW, WHO, WHEN, WHICH, WHERE, WHY and OTHER.[6] We evaluate the precision and recall of each question types. Figure 2 provides the precision and recall metrics of different question types. The precision and recall of a question type T are defined as:

$$\text{precision(T)} = \frac{\#(\text{true T-type questions})}{\#(\text{generated T-type questions})} \tag{10}$$

$$\text{recall(T)} = \frac{\#(\text{true T-type questions})}{\#(\text{all gold T-type questions})} \tag{11}$$

---

[6] We treat questions 'what country', 'what place' and so on as WHERE type questions. Similarly, questions containing 'what time', 'what year' and so forth are counted as WHEN type questions.

For the majority question types, WHAT, HOW, WHO and WHEN types, our NQG++ model performs well for both precision and recall. For type WHICH, it can be observed that neither precision nor recall are acceptable. Two reasons may cause this: a) some WHICH-type questions can be asked in other manners, e.g., 'which team' can be replaced with 'who'; b) WHICH-type questions account for about 7.2% in training data, which may not be sufficient to learn to generate this type of questions. The same reason can also affect the precision and recall of WHY-type questions.
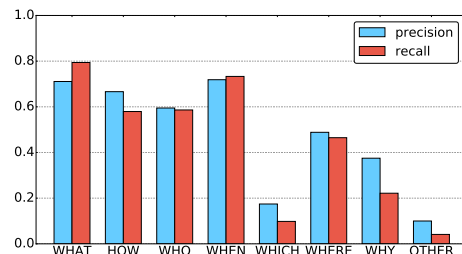

Fig. 2. Precision and recall of question types.

4 Conclusion and Future Work

In this paper we conduct a preliminary study of natural language question generation with neural network models. We propose to apply neural encoder-decoder model to generate answer focused questions based on natural language sentences. The proposed approach uses a feature-rich encoder to encode answer position, POS and NER tag information. Experiments show the effectiveness of our NQG method. In future work, we would like to investigate whether the automatically generated questions can help to improve question answering systems.

References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: Proceedings of 3rd International Conference for Learning Representations. San Diego (2015)
2. Chali, Y., Hasan, S.A.: Towards topic-to-question generation. Comput. Linguist. 41(1), 1–20 (Mar 2015)
3. Chen, D., Manning, C.: A fast and accurate dependency parser using neural networks. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 740–750. Association for Computational Linguistics, Doha, Qatar (October 2014)

4. Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder–decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1724–1734. Association for Computational Linguistics, Doha, Qatar (October 2014)

5. Fleiss, J.L.: Measuring nominal scale agreement among many raters. Psychological bulletin 76(5), 378 (1971)

6. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Aistats. vol. 9, pp. 249–256 (2010)

7. Goodfellow, I.J., Warde-Farley, D., Mirza, M., Courville, A.C., Bengio, Y.: Maxout networks. ICML (3) 28, 1319–1327 (2013)

8. Gulcehre, C., Ahn, S., Nallapati, R., Zhou, B., Bengio, Y.: Pointing the unknown words. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 140–149. Association for Computational Linguistics, Berlin, Germany (August 2016)

9. Heilman, M.: Automatic factual question generation from text. Ph.D. thesis, Carnegie Mellon University (2011)

10. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. In: Proceedings of 3rd International Conference for Learning Representations. San Diego (2015)

11. Luong, T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. pp. 1412–1421. Association for Computational Linguistics, Lisbon, Portugal (September 2015)

12. Manning, C.D., Schütze, H., et al.: Foundations of statistical natural language processing, vol. 999. MIT Press (1999)

13. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S.J., McClosky, D.: The Stanford CoreNLP natural language processing toolkit. In: Association for Computational Linguistics (ACL) System Demonstrations. pp. 55–60 (2014)

14. Nallapati, R., Zhou, B., glar Gulçehre, Ç., Xiang, B.: Abstractive text summarization using sequence-to-sequence rnns and beyond. In: Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning (2016)

15. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th annual meeting on association for computational linguistics. pp. 311–318. Association for Computational Linguistics (2002)

16. Pascanu, R., Mikolov, T., Bengio, Y.: On the difficulty of training recurrent neural networks. ICML (3) 28, 1310–1318 (2013)

17. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Empirical Methods in Natural Language Processing (EMNLP). pp. 1532–1543 (2014)

18. Rajpurkar, P., Zhang, J., Lopyrev, K., Liang, P.: Squad: 100,000+ questions for machine comprehension of text. arXiv preprint arXiv:1606.05250 (2016)

19. Serban, I.V., García-Durán, A., Gulcehre, C., Ahn, S., Chandar, S., Courville, A., Bengio, Y.: Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. In: Proceedings of ACL 2016. pp. 588–598. Association for Computational Linguistics, Berlin, Germany (August 2016)

20. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 15(1), 1929–1958 (2014)

21. Wang, S., Jiang, J.: Machine comprehension using match-lstm and answer pointer. arXiv preprint arXiv:1608.07905 (2016)