Leveraging Context Information for Natural Question Generation

Linfeng Song[1]*, **Zhiguo Wang**[2], **Wael Hamza**[2], **Yue Zhang**[3] **and Daniel Gildea**[1]
[1]Department of Computer Science, University of Rochester, Rochester, NY 14627
[2]IBM T.J. Watson Research Center, Yorktown Heights, NY 10598
[3]Singapore University of Technology and Design

Abstract

The task of natural question generation is to generate a corresponding question given the input passage (fact) and answer. It is useful for enlarging the training set of QA systems. Previous work has adopted sequence-to-sequence models that take a passage with an additional bit to indicate answer position as input. However, they do not explicitly model the information between answer and other context within the passage. We propose a model that matches the answer with the passage before generating the question. Experiments show that our model outperforms the existing state of the art using rich features.

1 Introduction

The task of natural question generation (NQG) is to generate a fluent and relevant question given a passage and a target answer. Recently NQG has received increasing attention from both the industrial and academic communities because of its values for improving QA systems by automatically increasing the training data. It can also be used for educational purposes such as language learning (Heilman and Smith, 2010).

One example is shown in Table 1, where a question "when was nikola tesla born ?" is generated given a passage and a fact "1856". Existing work for NQG uses a sequence-to-sequence model (Sutskever et al., 2014), which takes a passage as input for generating a question. They either entirely ignore the target answer (Du et al., 2017), or directly hard-code answer positions (Zhou et al., 2017; Yang et al., 2017; Subramanian et al., 2017; Tang et al., 2017; Wang et al., 2017a; Yuan et al., 2017). These methods can neglect rich potential interactions between the passage and the target answer. In addition, they fail when the target answer does not occur in the passage verbatim. In Table 1 the answer "1856" is the year when nikola tesla was born. This can be easily determined by leveraging the contextual information of "10 july 1856 – 7 january 1943", while it is relatively hard when only the answer position information is adopted.

Passage: nikola tesla (serbian cyrillic : Никола Тесла ; 10 july 1856 – 7 january 1943) was a serbian american inventor , electrical engineer , mechanical engineer , physicist , and futurist best known for his contributions to the design of the modern alternating current (ac) electricity supply system .
Question: when was nikola tesla born ?

Table 1: A QG example, where answer is underlined.

We investigate explicit interaction between the target answer and the passage, so that contextual information can be better considered by the encoder. In particular, matching is used between the target answer and the passage for collecting relevant contextual information. We adopt the multi-perspective context matching (MPCM) algorithm (Wang et al., 2017b), which takes two texts as input before producing a vector of numbers, representing similarity under different perspectives.

Results on SQuAD (Rajpurkar et al., 2016) show that our model gives better BLEU scores than the state of the art. Furthermore, the questions generated by our model help to improve a strong extractive QA system. Our code is available at https://github.com/freesunshine0316/MPQG.

* Work done during an internship at IBM.

Proceedings of NAACL-HLT 2018, pages 569–574
New Orleans, Louisiana, June 1 - 6, 2018.

2 Baseline: sequence-to-sequence

Our baseline is a sequence-to-sequence model (Bahdanau et al., 2015) with the copy mechanism (Gulcehre et al., 2016; Gu et al., 2016). It uses an LSTM encoder to encode a passage and an LSTM decoder to synthesize a question.

2.1 Encoder

The encoder is a bi-directional LSTM (Hochreiter and Schmidhuber, 1997), whose input x_j at step j is $[e_j; b_j]$, the concatenation of the current word embedding e_j with additional bit b_j indicating whether it belongs to the answer.

2.2 Decoder with the copy mechanism

The decoder is an attentional LSTM model, with the attention memory H being the concatenation of all encoder states. Each encoder state h_j is the concatenation of two bi-directional LSTM states:

$$h_j = [\overleftarrow{h_j}; \overrightarrow{h_j}] \quad (1)$$

$$H = [h_0; \ldots; h_N], \quad (2)$$

where N is the number of encoder states. At each step t, the decoder state s_t and context vector c_t are generated from the previous decoder state s_{t-1}, context vector c_{t-1} and output x_{t-1} in the same way as Bahdanau et al. (2015). The output distribution over a vocabulary is calculated via:

$$P_{vocab} = \text{softmax}(V_1[s_t; c_t] + b_1),$$

where V_1 and b_1 are model parameters, and the number of rows in V_1 is the size of the vocabulary.

Since many passage words also appear in the question, we adopt the copy mechanism (Gulcehre et al., 2016; Gu et al., 2016), which integrates the attention over input words into the final vocabulary distribution. The probability distribution is defined as the interpolation:

$$P_{final} = g_t P_{vocab} + (1 - g_t) P_{attn},$$

where g_t is the switch for controlling generating a word from the vocabulary or directly copying it from the passage. P_{vocab} is the vocabulary probability distribution as defined above, and P_{attn} is calculated based on the current attention distribution by merging probabilities of duplicated words. Finally, g_t is defined as:

$$g_t = \sigma(w_c^T c_t + w_s^T s_t + w_x^T x_{t-1} + b_2),$$

where the vectors w_c, w_s, w_x and scalar b_2 are model parameters.

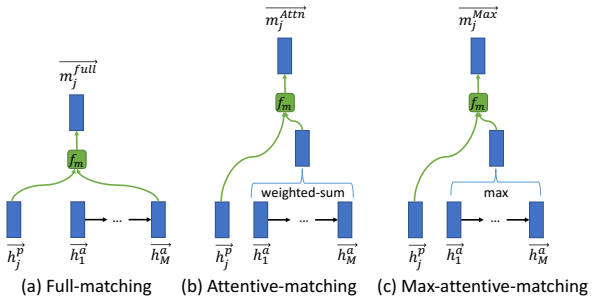

Figure 1: Matching strategies.

3 Method

Our model follows the baseline encoder-decoder framework. The encoder reads a passage $P = (p_1, \ldots, p_N)$ and an answer $A = (a_1, \ldots, q_M)$; the decoder generates a question $Q = (q_1, \ldots, q_L)$ word by word.

3.1 Multi-perspective encoder

Different from the baseline, we first encode both the passage and the answer by using two separate bi-directional LSTMs:

$$h_j^p = [\overleftarrow{h_j^p}, \overrightarrow{h_j^p}] = \text{BiLSTM}(\overleftarrow{h_{j+1}^p}, \overrightarrow{h_{j-1}^p}, p_j)$$

$$h_i^a = [\overleftarrow{h_i^a}, \overrightarrow{h_i^a}] = \text{BiLSTM}(\overleftarrow{h_{i+1}^a}, \overrightarrow{h_{i-1}^a}, a_i)$$

We use the multi-perspective context matching algorithm (Wang et al., 2017b) on top of the BiLSTM outputs, matching each hidden state h_j^p of the passage against all hidden states $h_1^a \ldots h_M^a$ of the answer. The goal is to detect whether each passage word belongs to the relevant context of the answer. Shown in Figure 1, we adopt three strategies to match the passage with the answer, each investigating different sources of information.

Full-matching considers the last hidden state of the answer, which encodes all words and the word order. *Attentive-matching* synthesizes a vector by computing a weighted sum of all answer states against the passage state, then compares the vector with the passage state. It also considers all words in the answer but without word order. Finally, *max-attentive-matching* only considers the most relevant answer state to the passage state.

Multi-perspective matching These strategies require a function f_m to match two vectors v_1 and v_2. It is defined as:

$$m = f_m(v_1, v_2; \mathbf{W}),$$

where $\mathbf{W}$ is a tunable weight matrix. Each row $W_k \in \mathbf{W}$ represents the weights associated with

one perspective, and the similarity according to that perspective is defined as:

$$m_k = \cos(W_k \odot v_1, W_k \odot v_2),$$

where $\odot$ is the element-wise multiplication operation. So $f_m(v_1, v_2; \mathbf{W})$ represents the matching results between v_1 and v_2 from all perspectives. Intuitively, each *perspective* calculates the cosine similarity between two reweighted input vectors, associated with a weight vector trained to highlight different dimensions of the input vectors. This can be regarded as considering a different part of the semantics captured in the vector.

The final matching vector m_j for the j-th word in the passage is the concatenation of the matching results of all three strategies. We employ another BiLSTM layer on top of the matching layer:

$$h_j^m = \overleftarrow{h_j^m}, \overrightarrow{h_j^m} = \text{BiLSTM}(\overleftarrow{h_{j+1}^m}, \overrightarrow{h_{j-1}^m}, m_j)$$

Comparison with the baseline The encoder states (h_j) of the baseline only contains the answer position information in addition to the passage content. The matching states (h_j^m) of our model includes the matching information of all passage words, and potentially contains the answer position information. The rich matching information can guide the decoder to generate more accurate questions.

3.2 Decoder with the copy mechanism

The decoder is identical to the one described in Section 2.2, except that matching information ($\overleftarrow{h_j^m}, \overrightarrow{h_j^m}$) is added to the attention memory:

$$h_j = [\overleftarrow{h_j^p}; \overrightarrow{h_j^p}; \overleftarrow{h_j^m}; \overrightarrow{h_j^m}] \quad (3)$$

$$H = [h_0; \ldots; h_N] \quad (4)$$

The attention memory contains not only the passage content, but also the matching information, which helps generate more accurate questions.

4 Experiments

Following existing work (Du et al., 2017; Zhou et al., 2017), experiments are conducted on the publically accessible part of SQuAD (Rajpurkar et al., 2016). The dataset contains 536 articles and over 100k questions, and around 10% is held by the organizer for fair evaluation.

Models	BLEU
M2S+cp	12.63
w/o full-matching	11.57
w/o attentive-matching	11.78
w/o max-attentive-matching	12.11

Table 2: Ablation results for matching strategies.

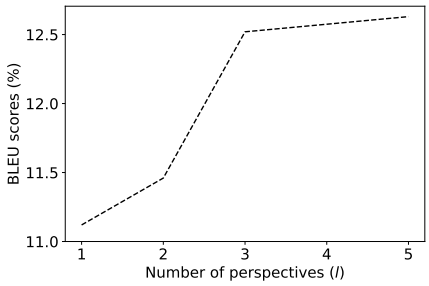

Figure 2: Effectiveness on the number of perspectives.

4.1 Settings

We evaluate our model for question quality against gold questions, as well as their effectiveness in improving an extractive QA system. Since Du et al. (2017) and Zhou et al. (2017) conducted their experiments using different training/dev/test split, we conduct experiments on both splits, and compare with their reported performance. For improving an extractive QA system, we use the data split of Du et al. (2017), and conduct experiments on low-resource settings, where only (10%, 20%, or 50%) of the human-labeled questions in the training data are available. We choose Wang et al. (2016) as the extractive QA system.

Both the baseline and our model are trained with cross-entropy loss. Greedy search is adopted for generating questions.

4.2 Development experiments

Matching strategies In Table 2, we analyze the importance of each matching strategy by performing an ablation experiment on the devset according to the data split of Du et al. (2017). We can see that there is a performance decrease when removing each of the three matching strategies, which means that all three strategies are complementary. In addition, *w/o max-attentive-matching* shows the least performance decrease. One likely reason is that *max-attentive-matching* considers only the most similar hidden state of the answer, while the other two consider all hidden states. Finally, *w/o full-matching* shows more performance decrease than *w/o attentive-matching*. A reason

Models	Split 1			Split 2
	BLEU	METEOR	ROUGE-L	BLEU
S2S-ans	12.28	16.62	39.75	–
S2S+cp+f	–	–	–	13.29
S2S+cp	12.22	17.38	39.03	12.59
M2S+cp	**13.98**	**18.77**	**42.72**	**13.91**

Table 3: Test results.

may be that *full-matching* captures word order information, while *attentive-matching* does not.

Number of perspectives Figure 2 shows the performance changes with different numbers of perspectives. There is a large performance improvement when increasing the number from 1 to 3, which becomes small when further increasing the number from 3 to 5. This shows that our multi-perspective matching algorithm is effective, as we do not need a large number of perspectives for reaching our reported performance.

4.3 Results

In Table 3, we compare our model with the previous state of the art: *S2S-ans* (Du et al., 2017) and *S2S+cp+f* (Zhou et al., 2017). Both methods use the sequence-to-sequence model. *S2S-ans* encodes only the passage, yet does not use answer position information. *S2S+cp+f* uses both answer position and rich features (NE and POS tags) by concatenating their embeddings with the word embedding on the encoder side (Peng et al., 2016), adopting the copy mechanism for their decoder. *S2S+cp* is our sequence-to-sequence baseline with the copy mechanism, and *M2S+cp* is our model, which further uses multi-perspective encoder.

M2S+cp outperforms *S2S+cp* on both data splits, showing that modeling contextual information is helpful for generating better questions. In addition, only taking word embedding features, *M2S+cp* shows better performance than *S2S+cp+f*. Both multi-perspective matching and rich features play a similar role of leveraging more information than the answer position information. However, *M2S+cp* can be applied to low-resource languages and domains, where there is not sufficient labeled data for training the taggers for generating rich features. *M2S+cp* is also free from feature engineering, which is necessary for *S2S+cp+f* on new domains.

Finally, unlike *S2S-ans*, *S2S+cp+f* and *S2S+cp*, *M2S+cp* can be useful when the answer is not explicitly contained in the passage, as it matches the target answer against the passage rather than using the answer position information.

Passage: nikola tesla (serbian cyrillic : Никола Тесла ; 10 july 1856 – 7 january 1943) was a serbian american inventor , electrical engineer , mechanical engineer , physicist , and futurist best known for his contributions to the design of the modern alternating current (ac) electricity supply system . **Reference:** when was nikola tesla born ? **S2S+cp:** when was nikola tesla 's inventor ? **M2S+cp:** when was nikola tesla born ?
Passage: zhèng (chinese : 正) meaning " right " , " just " , or " true " , would have received the mongolian adjectival modifiers , creating " jenggis " , which in medieval romanization would be written " genghis " . **Reference:** what does zhèng mean ? **S2S+cp:** what are the names of the " jenggis " ? **M2S+cp:** what does zhèng mean ?
Passage: the university of chicago (uchicago , chicago , or u of c) is a private research university in chicago . **Answer:** in illinois **M2S+cp:** where is the university of chicago located ?

Table 4: QG example, where answers are underlined.

4.4 Example Output

Table 4 shows example outputs of *M2S+cp* and *S2S+cp*. For the first case, *M2S+cp* recognizes that "1856" is the year when "nikola tesla" is born, while *S2S+cp* fails to. The matching algorithm of *M2S+cp* gives high matching numbers for the phrase "10 july 1856 – 7 january 1943" with "1856" having the highest matching number, while *S2S+cp* only highlights "1856". Simply highlighting "1856" can be ambiguous, while recognizing a pattern "day month year – day month year" with the first year being the answer is more definite. It is similar in the second case, where *M2S+cp* recognize "zhèng meaning right", which fits into the pattern "A meaning B" with B being the answer.

The third example is a case where the answer is not explicitly contained in the passage.[1] *M2S+cp* generates a precise question, even though the answer "in illinois" does not appear in the passage. On the other hand, *S2S+cp* fails in this case, as the answer position information can not be obtained from the input.

4.5 Question generation for extractive QA

Table 5 shows data augmentation results for extractive QA, where the gold questions of only a part of the training data are available. *Only-gold* uses only the available gold questions to train the

[1] This is modified from SQuAD, as all the original answers in the SQuAD dataset are explicitly contained in the passage.

Methods	Exact Match (EM)			F1 score		
	10%	20%	50%	10%	20%	50%
only-gold	47.87	57.98	63.60	59.64	68.05	73.02
S2S+cp	57.80	60.26	64.79	67.01	69.71	73.74
M2S+cp	**59.11**	**61.40**	**65.95**	**67.73**	**70.60**	**75.08**

Table 5: Results on improving extractive QA with automatically generated questions.

extractive QA model, while *S2S+cp* and *M2S+cp* use all training data, adopting the model-generated questions if the gold question is not available. For evaluation metrics, *F1 score* treats the prediction and ground-truth answer as bags of tokens, and compute their F1 score; *Exact Match* measures the percentage of predictions that match the ground truth answer exactly (Rajpurkar et al., 2016).

M2S+cp is consistently better than *S2S+cp* both under F1 score and Exact Match, showing that contextual information helps to generate more accurate questions. Besides, using 10% gold data, the automatically generated questions from *M2S+cp* help to reach a better performance than that using only 20% gold data, and it is 11 points better than that using only 10% gold data.

5 Conclusion

We demonstrated that natural question generation can benefit from contextual information. Leveraging a multi-perspective matching algorithm, our model outperforms the existing state of the art, and our automatically generated questions help to improve a strong extractive QA system.

Acknowledgments

We would like to thank the anonymous reviewers for their feedback.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR 2015*.

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 (ACL 2017)*. pages 1342–1352.

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

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*. Berlin, Germany.

Michael Heilman and Noah A. Smith. 2010. Good Question! Statistical ranking for question generation. In *The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-2010)*. pages 609–617.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Xi Peng, Rogerio S Feris, Xiaoyu Wang, and Dimitris N Metaxas. 2016. A recurrent encoder-decoder network for sequential face alignment. In *European Conference on Computer Vision*. pages 38–56.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of EMNLP 2016*. Austin, Texas, pages 2383–2392.

Sandeep Subramanian, Tong Wang, Xingdi Yuan, and Adam Trischler. 2017. Neural models for key phrase detection and question generation. *arXiv preprint arXiv:1706.04560* .

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

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

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

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017b. Bilateral multi-perspective matching for natural language sentences. In *IJCAI 2017*.

Zhiguo Wang, Haitao Mi, Wael Hamza, and Radu Florian. 2016. Multi-perspective context matching for machine comprehension. *arXiv preprint arXiv:1612.04211* .

Zhilin Yang, Junjie Hu, Ruslan Salakhutdinov, and William Cohen. 2017. Semi-supervised QA with generative domain-adaptive nets. In *Proceedings of the 55th Annual Meeting of the Association for*

Computational Linguistics (ACL-2017). Vancouver, Canada, pages 1040–1050.

Xingdi Yuan, Tong Wang, Caglar Gulcehre, Alessandro Sordoni, Philip Bachman, Saizheng Zhang, Sandeep Subramanian, and Adam Trischler. 2017. Machine comprehension by text-to-text neural question generation. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*. Vancouver, Canada, pages 15–25.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017. Neural question generation from text: A preliminary study. *arXiv preprint arXiv:1704.01792* .