Triple-Joint Modeling for Question Generation using Cross-Task Autoencoder

Hongling Wang[1], Renshou Wu[1], Zhixu Li[1,2], Zhongqing Wang[1], Zhigang Chen[3], and Guodong Zhou[1]

[1] School of Computer Science and Technology, Soochow University, Suzhou, China
[2] IFLYTEK Research, Suzhou, China
[3] State Key Laboratory of Cognitive Intelligence, iFLYTEK, P.R. China
{hlwang,zhixuli,gdzhou}@suda.edu.cn, rswu@stu.suda.edu.cn,
wangzq.antony@gmail.com zgchen@iflytek.com

Abstract. Question Generation (QG) aims to generate a question based on the context. Given the intrinsic connections between QG and QA (Question Answering), we focus on training a joint model for both QG and QA, and take one step further to integrate one more context self-encoding (CSE) task into the joint model, as a junction auxiliary task to better integrate QG and QA. In particular, our model employs a cross-task autoencoder to incorporate QG, QA and CSE into a joint learning process, which could better utilize the correlation between the contexts of different tasks in learning representations and provide more task-specific information. Experimental results show the effectiveness of our triple-task training model for QG, and the importance of learning interaction among QA and CSE for QG.

Keywords: Question Generation · Autoencoder · Joint learning.

1 Introduction

Question Generation (QG) is an important task in machine comprehension and perception of knowledge, which has also aroused a lot of attention in cognitive science and artificial intelligence during the last decade. Recently, some studies [17, 15] consider the interaction between QG and QA, leveraging their probabilistic correlation to guide the learning process. Among them, [15] uses the "duality" between QA and QG as a regularization term to influence the learning of QA and QG models. Both [17, 15] report that the co-training models could improve both QA and QG performance. But in most cases, existing work on joint learning for QG and QA which learns the correlation between two uni-task models with a shared encoding layer [17]. However, a shared encoding layer tends to learn common information between tasks, but might ignore those information that is only useful for one task, which we call as task-specific information.

In this paper, we take one step further to integrate one more context self-encoding (CSE) task into the joint model, as a junction auxiliary task to better

integrate QG and QA. CSE is an autoencoder, which could provide more complete information(i.e., both common information and task-specific information) for QA and QG. In particular, our model incorporates QG, QA and CSE into a single learning process with a so-called cross-task autoencoder. The proposed cross-task autoencoder reconstructs one input from the middle representations of other different tasks. For example, it reconstructs context from the intermediate representations of QG and QA task, that could better utilize the correlation between the context of different tasks in representation learning and provide more task-specific information. Experimental results show that our Triple-Joint Model yields much better performance over several state-of-the-art models.

2 Related Work

Typically, QG could be processed by transforming input text into symbolic representation, which then will be transformed into a question. The traditional methods for QG are either based on rules [1], or slot-filling with question templates [7], which often involve pipelines of independent components that are difficult to be tuned for final performance measures. sTo address the weaknesses above, some end-to-end neural models are proposed. For example, [4] use a sequence-to-sequence model with an attention mechanism derived from the encoder states.

Given the intrinsic connections between QA and QG, some recent efforts propose to train a joint model for both QA and QG. [17] take both QA and QG as generation tasks and adopt an attention-based sequence-to-sequence architecture. Alternatively, [15] use the "duality" between QA and QG as a regularization term to influence the learning of QA and QG models. They implemente simple yet effective QA and QG models, both of which are neural network based approaches.

As stated by [12], the objective and its importance are the two core aspects of a question. We could hardly judge the quality of a question without knowing the context of the question. Thus, people tend to introduce context information into a QG system to guide the generation of questions, instead of using the input answer only. To the best of our knowledge, we are the first to do triple-joint learning for QA and QG models by introducing one more task, that is, context self-encoding (CSE).

3 Basic Uni-Task Model

In this section, we first give a general definition of the QG task, then introduce our uni-task model that could be applied for QG task alone.

3.1 Tasks Definition

The primary task in this paper is QG, which can be formally defined as follows:

Given an answer $A = (w_1^a, w_2^a, ...w_{T_A}^a)$, and relevant sequential sentences $C = (w_1^c, w_2^c, ..., w_{T_C}^c)$ as input context, the task is to generate a sequential text, say question $Q = (w_1^q, w_2^q, ..., w_{T_Q}^q)$, that could be answered by the answer A according to C, where T_x denotes the length of x ($x \in \{C, Q, A\}$).

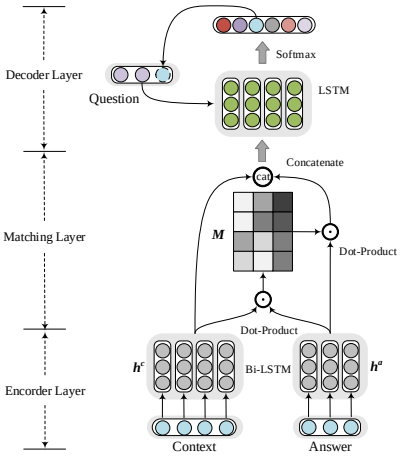

Fig. 1. Uni-Task Model Overview

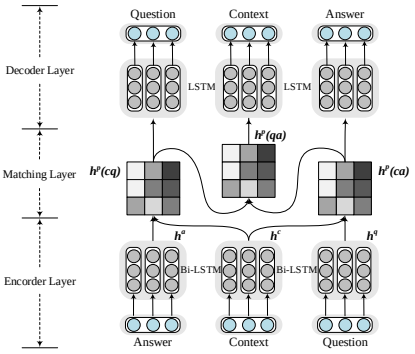

Fig. 2. Triple-Joint Model Overview

3.2 Overview of Uni-Task Model

Figure 1 depicts the overview of our uni-task model for QG task, which can be separated into three parts: encoder layer, dot-product matching layer and decoder layer.

3.3 Encoder Layer

We first convert the words x_t into continuous representations with an embedding matrix W_e. After that, the input sequence is processed by a bidirectional recurrent network [8].

$$\overrightarrow{h_t} = \overrightarrow{f}(x_t, h_{t-1}), \tag{1}$$

$$\overleftarrow{h_t} = \overleftarrow{f}(x_t, h_{t+1}) \tag{2}$$

$$h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}] \tag{3}$$

where $h_t \in \mathbb{R}^n$ is a hidden state at time t. $f(\cdot)$ is LSTM.

3.4 Dot-product Matching Layer

In order to incorporate answer information into context representation, We introduce a match mechanism to decide the importance of each individual representation automatically, using Dot-Product to calculate a pair-wise matching matrix M which indicates the pair-wise matching degree of one context word and one answer word.

Formally, when given context and answer representation h^c and h^a, we can compute a context and answer pair representation h^p as follows:

$$h_t^p = f(h_{t-1}^p, [h_t^c, c_t]) \tag{4}$$

where c_t is an attention-pooling vector of the whole answer, which can be computed as a weighted sum of the sequence of annotations $(h_1^a, \cdots, h_{T_a}^a)$ to which an encoder maps the answer, that is,

$$c_t = \frac{\sum_{i=1}^{T_a} exp(M(t,i))h_i^a}{\sum_{k=1}^{T_a} exp(M(t,k))} \tag{5}$$

where $M(t,i)$ is the weight of annotation h_i^a in time step t, which is the pair-wise matching degree of one context word and one answer word, i.e.,

$$M(t,i) = \frac{h_t^c (h_i^a)^T}{\sqrt{d_k}} \tag{6}$$

where d_k is the dimension of h^c.

3.5 Decoder Layer

We implement a attention-based LSTM decoder to read the context-answer pair representation h^p and generate question word by word. At each time step, the decoder generates a question word y_i by sampling from a distribution of the target vocabulary until sampling the token representing the end of sentence. The hidden state of the decoder s_i and the pair representation h_i^p at each time step i of the encoding process are computed with a weight matrix W_α to obtain the global attention $\alpha_{i.j}$ and the context vector c_i. It is described below:

$$p_{gen}(y_i|y_1, ..., y_{i-1}, \mathbf{x}) = g(y_{i-1}, s_i, c_i) \tag{7}$$

$$s_i = f(s_{i-1}, y_{i-1}, c_i) \tag{8}$$

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j^p \tag{9}$$

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})} \tag{10}$$

$$e_{ij} = a(s_{i-1}, h_j^p) \tag{11}$$

where $f(\cdot)$ is LSTM, $g(\cdot)$ is a nonlinear, potentially multi-layered function that outputs the probability of y_t, and $a(\cdot)$ is a feedforward neural network.

3.6 Training and Inference

Given the context c and answer a, our uni-task model for QG can be trained end-to-end using stochastic gradient descent by minimizing the negative conditional log likelihood of the reference y with respect to θ:

$$\mathcal{L}_{single} = -\sum_{t=1}^{T} \log p(y_t|y_{<t}, c, a; \theta) \tag{12}$$

where θ represents the trainable model parameters.

4 Triple-Joint Modeling using Cross-Task Autoencoder

There are some existing work on joint learning for QG and QA which learns the correlation between two uni-task models through a shared encoding layer [16, 17]. However, a shared encoding layer tends to learn common information between tasks, but might ignore the task-specific information.To address above limitations, we propose a Triple-Joint model for QG by integrating with QA and CSE. In particular, our joint model incorporates QG, QA and CSE into a single training process with a cross-task autoencoder.

4.1 Cross-Task Autoencoder

Autoencoder is able to encode texts in a way to better preserve syntactic, semantic, and discourse coherence [5]. However, the basic autoencoder model is not suitable for exploiting the complex correlations of representations from different tasks, given that it will learn different level representation for inputs with different reconstruction losses.

Meanwhile, considering that the context is the core junction to connect QG and QA task, while QG and QA tasks are good choices to guide the autoencoder to learn the context representation, we integrate a context self-encoding task(CSE) into QA&QG joint model and propose a cross-task autoencoder to learn complete context information (i.e., both common information and task-specific information) for different task in order to alleviate the weak point of shared layer in traditional joint learning. Unlike the basic autoencoder which reconstructs the input itself, this cross-task autoencoder reconstructs input from the intermediate representations of different tasks. It incorporates representation learning and correlation learning into a single process, thus some correlations of QG, QA and CES could be captured in the reconstruction loss.

As illustrated in Figure 2, our triple-joint model with cross-task autoencoder can be viewed as a combination of three sub-networks, each of which is a basic uni-task model. Among them, CSE task is the core junction to connect QG task and QA task. These sub-networks are connected by the intermediate representation of context. Each sub-network in the triple-joint model is responsible for a

task. In the learning process, the three sub-networks are coupled at their intermediate representation of matching layer. After learning, the three sub-networks could exhibit the corresponding complete representation.

For our triple-joint cross-task autoencoder, both the input x_1, x_2, x_3 and output y_1, y_2, y_3 are the same context, question and answer. The model first encoding context, question and answer into vector representation h^c, h^q and h^a, then reconstructs question based on the match representation $h^p(c,a)$ of h^c and h^a, as well as the answer based on match representation $h^p(c,q)$ of h^c and h^q. Meanwhile, the match representation $h^p(a,q)$ of $h^p(c,a)$ and $h^p(c,q)$ is used to reconstruct context, different from $h^p(c,a)$ and $h^p(c,q)$, $h^p(a,q)$ is computed as follows:

$$\tilde{h^p}(a,q) = tanh([h^p(c,a); h^p(c,q)]W_r) \tag{13}$$

$$gate = \sigma([h^p(c,a); h^p(c,q)]W_g) \tag{14}$$

$$h^p(a,q) = gate \cdot \tilde{h^p}(a,q) + (1 - gate) \cdot h^c \tag{15}$$

where the projections are parameter matrices $W_r \in \mathbb{R}^{2d_{model} \times d_{model}}$, $W_g \in \mathbb{R}^{2d_{model} \times d_{model}}$, $\sigma(\cdot)$ denotes the sigmoid activation function, the output vector $h^p(a,q)$ is a linear interpolation of the input h^c and the intermediate vector $\tilde{h^p}(a,q)$. A *gate* is used to control the composition degree to which the intermediate vector is exposed.

4.2 Training and Inference

Given a training corpus include context c, question q and answer a, the Cross-Task Autoencoder can also be trained end-to-end. The loss function consists three parts: the loss computed by QG task, QA task and CES task respectively. To simplify the notations, the network parameters are grouped as θ.

$$\begin{aligned} \mathcal{L}_{Joint} = \lambda_1 \mathcal{L}_{single}(q'; c, a; \theta_q) + \lambda_2 \mathcal{L}_{single}(a'; c, q; \theta_a) \\ + (1 - \lambda_1 - \lambda_2) \mathcal{L}_{single}(c'; c, a, q; \theta_c) \end{aligned} \tag{16}$$

where λ_1 and λ_2 are hyper-parameters to tune the impacts of the QG and QA tasks. Meanwhile, it is trade off between two groups of objectives: correlation losses and reconstruction losses. An appropriate value for λ_1 and λ_2 are crucial.

5 Experiments

5.1 Experimental Settings

The experimental settings include dataset, hyper-parameters and evaluation metrics.

Dataset. We mainly focus on the the Stanford Question Answering Dataset (SQuAD) [11] processed by [4] to train and evaluate our model. Since the dataset proessed by [4] lacks the answer corresponding to the question, we seek the

Table 1. Automatic Evaluation Results of Different Models

Model	BLEU 1	BLEU 2	BLEU 3	BLEU 4	METEOR	ROUGE-L
Seq2seq	31.34	13.79	7.36	4.26	9.88	29.75
Att-Seq2seq	43.09	25.96	17.50	12.28	16.62	39.75
MPQG	-	-	-	12.84	18.02	41.39
MPQG+R	-	-	-	13.98	18.77	42.72
JointQA	44.72	27.21	18.43	13.05	17.94	42.76
Triple-Joint CAE	**46.72**	**29.34**	**20.40**	**14.76**	**19.12**	**44.05**

corresponding document of question from document-list provided by [4], which is used to find the corresponding answer to expand the dataset.

Hyper-paramters. We implement our experiments in PyTorch on an NVIDIA Tesla V100 GPU. For word embedding, we use pre-trained case-sensitive GloVe embeddings [10] for both contexts, questions and answers. The number of LSTM hidden units (h_{model}) is 600 and we set the number of LSTMs layers to 2 in both the encoder and the decoder. $\lambda_1 = \lambda_2 = 0.4$

Evaluation Metrics. Following the previous studies [4], we choose the evaluation package released by [2] to evaluate the performance of our model, which was originally used to score image captions. The package includes BLEU 1, BLEU 2, BLEU 3, BLEU 4 [9], METEOR [3] and ROUGE-L [6] evaluation scripts.

5.2 Comparison with Baselines

To evaluate the performance of our model, we compare our model with several state-of-the-art QG methods as listed below, where we directly adopt the experimental settings and results reported by [4] for the first 5 baselines in the list.

- Seq2seq [14] is a basic encoder-decoder sequence learning system.
- Att-Seq2Seq [4] using a conditional neural language model with a global attention mechanism, they entirely ignored the answer.
- MPQG [13] follows the classic encoder-decoder framework. The encoder takes a passage and an answer as input then performs answer understanding by matching the answer with the passage from multiple perspectives.
- MPQG+R [13] is the model developed from MPQG. Here the model is fine-tuned with the policy gradient reinforcement learning algorithm after pre-training.
- JointQA [17] is a QG and QA joint model, which encodes the document and generates a question (answer) given an answer (question).

Table 1 shows automatic metric evaluation results for our models and baselines. As can be observed, our triple-joint model with cross-task autoencoder (**Triple-Joint CAE** for short in the tables) reaches better performances on

Table 2. Influence of Different Auxiliary Tasks

Model	BLEU 1	BLEU 2	BLEU 3	BLEU 4	METEOR	ROUGE-L
Uni-Task Model	45.11	27.89	19.21	13.76	18.41	43.61
QG&CES-Joint Model	45.32	28.14	19.32	13.88	18.51	43.72
QG&QA-Joint Model	45.78	28.62	19.71	14.18	18.73	43.85
Triple-Joint CAE	**46.72**	**29.34**	**20.40**	**14.76**	**19.12**	**44.05**

Table 3. Examples of generated questions, where the golden answers are underlined, and the text copied from context are in italian style in the generated questions.

Context	In the First World War, Devonport was the headquarters of Western Approaches Command until 1941 and *Sunderland flying boats* were operated by the Royal Australian Air Force.
Golden	What force used Sunderland flying boats out of Devonport?
Uni-Task	Who operated the *Sunderland boats*?
Triple-Joint	Who operated the *Sunderland flying boats*?

QG tasks than the baseline models. In particular, compared with the JointQA model, our model has 1.71, 1.18 and 1.29 relative gain in BLEU4, METEOR and ROUGE-L respectively, which proves that our auxiliary task CSE makes some contributions to our primary tasks, and the effectiveness of the employed cross-task autoencoder in joint learning.

5.3 Influence of Different Auxiliary Tasks

We compared the performance of our proposed model with Uni-Task Model and Joint model to further illustrate the influence of the QA task and CSE task for QG task.

The experimental results are shown in Table 2. We can observe that: (1) Our Uni-Task model could outperform the state-of-the-art MPQG model and the JoinQA model, which demonstrates that our dot-product matching strategy, although simple, works effectively in the model. (2) Both the QG&CES-Joint model and QG&QA-joint model outperform the Uni-Task model, which proves that joint models can utilize the intrinsic connections between the two tasks, and learn more useful information. (3) Compared with the Uni-Task model and the Joint models, our Triple-Joint CAE model performs better. This indicates that cross-task autoencoder could better utilize the correlation between the context of different tasks in representation learning, so this CSE task could provide more complete information for QG.

5.4 Discussion and Analysis

In this subsection, we provide an examples and analyze the performance of different question types.

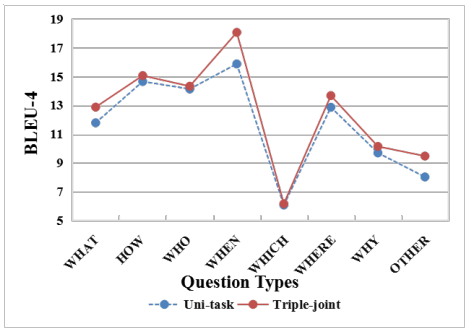

Fig. 3. Comparison by question types.

Case Study. Table 3 lists an example for case study. Basically, our model could generate questions that are semantically similar to the golden questions. For example, it can be observed that our model tends to use the text related to the answer from the context to generate questions, which are more semantically consistent with the context and the target answers.

Analysis on Type of Generated Questions. We classify the questions into different types, i.e., WHAT, HOW, WHO, WHEN, WHICH, WHERE, WHY and OTHER, and evaluate the generated questions and answers for each question type. Figure 3 shows the automatic metric evaluation results of different question types. As can be seen, the Triple-Joint CAE model always performs better than the Uni-Task model, while the best performance can be observed on the WHEN problems. For the majority question types, WHAT, HOW, WHO and WHEN, our model performs well. For type WHICH, it can be observed that neither precision nor recall are acceptable. The reason may cause this: WHICH-type questions account for about 7.2% in training data, which may not be sufficient to learn to generate this type of questions.

6 Conclusion

This work incorporates QG, QA and CSE into a single training process with cross-task autoencoder for joint representation learning. Experiments conducted on processed SQuAD dataset show that our triple-joint model outperforms several state-of-the-art QG models we compare with in this paper. As a future work, we will consider using Variational Autoencoder (VAE) to generate more semantically reasonable questions, which has been shown successful on several areas such as image generation.

Acknowledgments

This research is partially supported by National Natural Science Foundation of China (Grant No. 61632016, 61572336, 61572335, 61772356), and the Nat-

ural Science Research Project of Jiangsu Higher Education Institution (No. 17KJA520003, 18KJA520010).

References

1. Ali, H., Chali, Y., Hasan, S.A.: Automation of question generation from sentences. In: Proceedings of QG2010: The Third Workshop on Question Generation. pp. 58–67 (2010)
2. Chen, X., Fang, H., Lin, T.Y., Vedantam, R., Gupta, S., Dollár, P., Zitnick, C.L.: Microsoft coco captions: Data collection and evaluation server. arXiv preprint arXiv:1504.00325 (2015)
3. Denkowski, M., Lavie, A.: Meteor universal: Language specific translation evaluation for any target language. In: Proceedings of the ninth workshop on statistical machine translation. pp. 376–380 (2014)
4. Du, X., Shao, J., Cardie, C.: Learning to ask: Neural question generation for reading comprehension. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). vol. 1, pp. 1342–1352 (2017)
5. Li, J., Luong, M.T., Jurafsky, D.: A hierarchical neural autoencoder for paragraphs and documents. arXiv preprint arXiv:1506.01057 (2015)
6. Lin, C.Y.: Rouge: A package for automatic evaluation of summaries. Text Summarization Branches Out (2004)
7. Lindberg, D., Popowich, F., Nesbit, J., Winne, P.: Generating natural language questions to support learning on-line. In: Proceedings of the 14th European Workshop on Natural Language Generation. pp. 105–114 (2013)
8. Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J., Khudanpur, S.: Recurrent neural network based language model. In: Eleventh Annual Conference of the International Speech Communication Association (2010)
9. 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)
10. Pennington, J., Socher, R., Manning, C.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). pp. 1532–1543 (2014)
11. Rajpurkar, P., Zhang, J., Lopyrev, K., Liang, P.: Squad: 100,000+ questions for machine comprehension of text. arXiv preprint arXiv:1606.05250 (2016)
12. Rus, V., Arthur, C.G.: The question generation shared task and evaluation challenge. In: The University of Memphis. National Science Foundation. Citeseer (2009)
13. Song, L., Wang, Z., Hamza, W.: A unified query-based generative model for question generation and question answering. arXiv preprint arXiv:1709.01058 (2017)
14. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in neural information processing systems. pp. 3104–3112 (2014)
15. Tang, D., Duan, N., Qin, T., Yan, Z., Zhou, M.: Question answering and question generation as dual tasks. arXiv preprint arXiv:1706.02027 (2017)
16. Wang, J., Tian, J., Qiu, L., Li, S., Lang, J., Si, L., Lan, M.: A multi-task learning approach for improving product title compression with user search log data. arXiv preprint arXiv:1801.01725 (2018)
17. Wang, T., Yuan, X., Trischler, A.: A joint model for question answering and question generation. arXiv preprint arXiv:1706.01450 (2017)