Teaching Machines to Ask Questions

Kaichun Yao[1], **Libo Zhang**[2,*], **Tiejian Luo**[1], **Lili Tao**[3], **YanJun Wu**[2]
[1] University of the Chinese Academy of Sciences
[2] Institute of Software Chinese Academy of Sciences
[3] University of the West of England
yaokaichun@outlook.com, libo@iscas.ac.cn,
tiejian@ucas.ac,cn, Lili.Tao@uwe.ac.uk, yanjun@iscas.ac.cn

Abstract

We propose a novel neural network model that aims to generate diverse and human-like natural language questions. Our model not only directly captures the variability in possible questions by using a latent variable, but also generates certain types of questions[1] by introducing an additional observed variable. We deploy our model in the generative adversarial network (GAN) framework and modify the discriminator which not only allows evaluating the question authenticity, but predicts the question type. Our model is trained and evaluated on a question-answering dataset *SQuAD*, and the experimental results shown the proposed model is able to generate diverse and readable questions with the specific attribute.

1 Introduction

Automatic question generation aims to generate natural questions from a given text passage and a target answer. It has many important application values: One potential value for question generation is in education, where the automatic tutoring systems generate natural questions for reading comprehension materials [Heilman and Smith, 2010]. In the dialogue system, question generation techniques can help the dialogue agents launch a conversation or request feedback [Li *et al.*, 2017a]. In the medical field, question generation systems are also used as a clinical tool for evaluating or improving mental health [Colby *et al.*, 1971]. As the reverse task of question answering, question generation also has the potential for generating large-scale question answer pair corpus[Serban *et al.*, 2016].

Traditional methods for question generation mainly use rigid heuristic rules to convert an input passage into corresponding questions [Heilman, 2011; Chali and Hasan, 2015]. However, these approaches strongly rely on human-designed transformation and generation rules so that cannot be easily adopted to other domains. Recently, neural networks based question generation methods aim at training an end-to-end system to generate natural language questions from text without the human-designed rules [Zhou *et al.*, 2017]. The work adapts the sequence-to-sequence approach [Cho *et al.*, 2014] for generating questions, in which the encoder encodes the text passage and other auxiliary information (answer or context), and then a decoder is used to sequentially output question words. However, the existing encoder-decoder models only generate a single question for one text passage.

Given a text passage and a question, a unique answer can be found in the passage, but multiple questions can be asked by giving a passage and an answer. Learning the variety of a valid question is an important but overlooked problem in many existing methods [Zhou *et al.*, 2017; Yuan *et al.*, 2017]. In order to capture the diversity in the potential questions, our method aims to produce a series of questions from a given passage and a given answer. Human ask different types of questions. They are classified into six types in *SQuAD* dataset [Rajpurkar *et al.*, 2016], - WHO/WHOM, WHAT, WHICH, HOW, WHEN, OTHER [2]. Our method also aims to learn the generation of the certain types of questions.

In this paper, we teach machines to ask the right questions - the natural language questions are generated by learning the given text passage and a targeted answer. The key idea of our work is to model the question generations as a one-to-many problem. Given a text passage and an answer, multiple valid questions may exist. We model a probabilistic distribution over the potential questions using a latent variable. This allows us to generate diverse questions by drawing samples from the learned distribution and reconstruct the words sequence via a decoder neural network. Meanwhile, the observed question-type labels represent the salient attributes of questions and are provided to the models as conditionals in order to generate questions with the certain types.

We apply adversarial training to natural-language question generation task, in which we simultaneously train a generative model G and a discriminative model D. We use the latent variable and the observation variable in G to learn a distribution over potential questions, generating diverse and certain types of questions. We also made a modification for D, which encourages G to produce sequences that are indistinguishable from the questions generated by the human, and

[*] Corresponding author: Libo Zhang (libo@iscas.ac.cn).

[1] We classify the questions into six types - WHO, WHAT, WHICH, HOW, WHEN and OTHER.

[2] Six-types questions account for about 11.1%, 57.7%, 7.1%, 10.5%, 6.3%, 7.3% in training data, respectively.

contain the specific attributes. The main contributions of this paper are as follows:

- We present a novel natural-language question generation model deployed in GAN [Goodfellow *et al.*, 2014] framework which consists of a generative model G and a discriminative model D. D encourages G to generate more readable and diverse questions with the certain types.
- The variational auto-encoders (VAE) [Kingma and Welling, 2013] is adopted to the generative model G by introducing a latent variable to capture question diversity.
- The question type is regarded as the salient attribute and the observed variable, which is used to learn a disentangled representation from the latent distribution to produce the certain types of questions.

We train and evaluate our model on *SQuAD* dataset, and the experimental results show our model is able to generate the certain types of questions with high readability and diversity.

2 Related Work

2.1 Question Generation

Automatic question generation has drawn an increasing attention from natural language generation community in recent years. Majority of earlier work uses a rule-based method that transforms a sentence into related questions [Heilman, 2011] by manually constructed template. However, these methods strongly depend on manually generated rules so that cannot be easily adopted to other domains. Instead of generating questions from texts, a neural network method is trained to convert knowledge base triples to generate factoid questions from structured data [Serban *et al.*, 2016]. More recently, neural networks based question generation models aim at training an end-to-end system to generate natural language questions from text without human-designed transformation and generation rules [Zhou *et al.*, 2017; Yuan *et al.*, 2017; Du *et al.*, 2017]. These work extends the sequence-to-sequence models [Cho *et al.*, 2014] by enriching the encoder with auxiliary feature information to help generating better sentence encoding. Then the decoder with attention mechanism [Bahdanau *et al.*, 2014] produces natural language question of sentence.

2.2 Deep Generative Models

Deep neural network based generative models are widely used in text generation tasks such as text summarization [See *et al.*, 2017], machine translation [Hu *et al.*, 2017] and dialogue system [Li *et al.*, 2017b]. Deep generative models have drawn a lot attentions. Recurrent neural networks (RNNs) based generative model is proposed in [Graves, 2013] to generate the next word conditioned on previous work sequence. Encoder-decoder architecture [Cho *et al.*, 2014] based generative models use RNNs to encode an input text sentence into a fixed vector, and generate a new output text sequence from the vector using the second RNN model. These models usually produce one corresponding text sequence from a given text sequence. Recently, VAE [Kingma and Welling, 2013] as a popular framework has been applied in text generation tasks such as dialogue system [Zhao *et al.*, 2017] and image caption [Rezende *et al.*, 2014] as deep generative models. VAE encodes the input sequence into a latent hidden space, and then utilizes a decoder network to rebuild the original input by sampling from this space, aiming to capture the variability in potential generated sequences. GAN [Goodfellow *et al.*, 2014], an alternative training approach to generative models, where the training procedure is a minimax two-player game, in which a generative model is trained to generate outputs, while the discriminative model evaluates them for authenticity. To apply GAN to text generation, SeqGAN [Yu *et al.*, 2016] models the text generation as a sequential decision making process, and utilizes policy gradient methods [Sutton *et al.*, 1999] to train the generative model.

3 Model

In this section, we introduce our neural question generation model. As depicted in figure 1, we apply adversarial training to natural-language question generation task, in which we simultaneously train a generative model G and a discriminative model D. Generative model is employed in an encoder-decoder architecture [Cho *et al.*, 2014]. It is based on conditional variational autoencoders [Zhao *et al.*, 2017] that captures the diversity in the encoder, introducing the latent variable z and the observation variable c in G in order to learn a distribution over potential questions given an answer. We also made a simple modification for D by adding an auxiliary classifier to distinguish the types of questions. Similar to the standard adversarial training manner [Goodfellow *et al.*, 2014], we first pre-train the generative model by generating questions given the passages and answers. Then we pre-train the discriminator by providing positive examples from the human-generated questions and the negative examples produced from the pre-trained generator. After the pre-training, the generator and discriminator are trained alternatively.

3.1 Generative Model

Model Description

Question generation task is to model the true probability of a question Y given an input text passage X and an answer A. We denote our model distribution by $P(Y|X,A)$. We introduce a latent variable z which is used to learn to the latent distribution over the valid questions. We then define the conditional distribution $P(Y,z|X,A) = P(Y|z,X,A)P(z|X,A)$ and our goal is to use deep neural networks (parametrized by θ) to approximate $P(z|X,A)$ and $P(Y|z,X,A)$. We refer to $P_\theta(z|X,A)$ as the prior network and $P_\theta(Y|z,X,A)$ as the question generation decoder. Then the generative process of Y can be depicted as: (1) Sample the latent variable z from the prior network $P_\theta(z|X,A)$. (2) Generate Y through the question generation decoder $P_\theta(Y|z,X,A)$.

We assume the latent variable z follow multivariate Gaussian distribution with a diagonal covariance matrix. At training time, we follow the variational autoencoder framework [Kingma and Welling, 2013; Sohn *et al.*, 2015] and introduce an approximation network $Q_\phi(z|X,A)$ to approximate the true posterior distribution $P_\theta(z|X,A)$. We thus have the

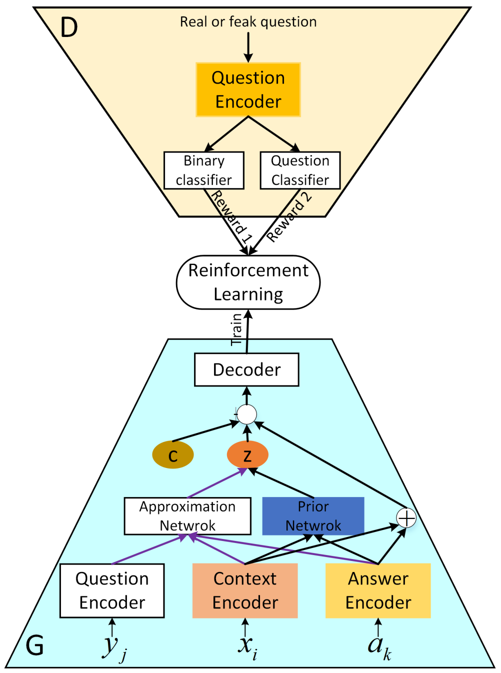

Figure 1: The neural question generation model deployed in GAN framework. $\oplus$ denotes the concatenation of the input vectors.

following evidence lower bound (ELBO) [Sohn *et al.*, 2015]:

$$\begin{aligned} logP(Y|X,A) \geq & -KL(Q_\phi(z|X,A)||P_\theta(z|X,A)) \\ & +E_{Q_\phi(z|X,A)}[logP_\theta(Y|z,X,A)] \end{aligned} \tag{1}$$

We point out that existing encoder-decoder models encode an input passage X and answer A as a single fixed representation. Hence, all of the possible questions corresponding to X and A must be stored within the decoder's probability distribution $P(Y|X,A)$, and during decoding it is hard to disentangle these possible questions. However, our question generation model contains a stochastic component z in the decoder $P(Y|z,X,A)$, and so by sampling different z and then performing maximum likelihood decoding on $P(Y|z,X,A)$, we hope to tease apart the questions stored in the probability distribution $P(Y|X,A)$.

In our question generation task, the generative model uses the latent variable z to learn the potential questions distribution and generate the diverse question by sampling different z. However, it is hard to produce the specific attributes or types of questions by randomly sampling from z. For instance, every question has its corresponding type and we hope to generate the specific types of questions. Inspired from AC-GAN [Odena *et al.*, 2017], we introduce another observed variable – question type label c, which is independent of z, to learn a disentangled representation from z. Our model can encode useful information into z and we want to tease apart these information by using an additional observed variable c. We update ELBO as follows:

$$\begin{aligned} logP(Y|X,A) \geq & -KL(Q_\phi(z|X,A)||P_\theta(z|X,A)) \\ & +E_{Q_\phi(z|X,A)}[logP_\theta(Y|(z,c),X,A)] \end{aligned} \tag{2}$$

Note that this loss decomposes into two parts: the KL divergence between the approximate posterior and the prior, and the cross-entropy loss between the model distribution and the data distribution. Finally, the model is trained by minimizing the following loss:

$$\begin{aligned} J_{ml}(\theta,\phi) = & KL(Q_\phi(z|X,A)||P_\theta(z|X,A)) \\ & -E_{Q_\phi(z|X,A)}[logP_\theta(Y|(z,c),X,A)] \end{aligned} \tag{3}$$

Model Implementation

Given an input text passage $X = \{x_1,\dots,x_n\}$, a question $Y = \{y_1,\dots,y_m\}$ and the corresponding answer $A = \{a_1,\dots,a_l\}$. Here, n, m and l are the length of the text passage, the length of ground truth question and the length of answer, respectively. Sequence elements x_i, y_j and a_k are given by pre-trained glove embedding vectors [Pennington *et al.*, 2014].

At the stage of encoding, firstly, we augment each document word embedding with a binary feature that indicates whether the passage word belongs to the answer. Then, we use a bidirectional RNNs [Schuster and Paliwal, 1997] with long short-term memory (LSTM) [Hochreiter and Schmidhuber, 1997] cell as the context encoder which runs on the augmented passage sequence, generating the corresponding hidden state vectors $h^d = \{h_1^d,...,h_n^d\}$ for the input tokens. Here, h_i^d is the concatenation of the RNNs' forward hidden state $\vec{h}_i^d$ and backward hidden state $\overleftarrow{h}_i^d$, i.e., $h_i^d = [\vec{h}_i^d;\overleftarrow{h}_i^d]$[3]. Meanwhile, we obtain the semantic representation of passage h^D by concatenating the last hidden states of the forward and backward RNNs, i.e., $h^D = [\vec{h}_n^d;\overleftarrow{h}_1^d]$.

We assume that the answer A consists of the sequence of words $\{x_s,...,x_e\}$ in the passage, where s and e are the start position and the end position of the answer word in the passage, s.t. $1 \leq s \leq e \leq n$. In order to obtain the semantic representation h^a of the answer A, we concatenate the hidden states $\{h_s^d,...,h_e^d\}$ of the context encoder corresponding to the answer word positions in the passage with the answer word embeddings $\{a_s,...,a_e\}$, i.e., $[h_i^d;a_i]$, $s \leq i \leq e$. We form h^a by calculating the average pooling of the concatenated answer representation of each position.

$$h^a = \frac{1}{e-s+1}\sum_{i=s}^{e}[h_i^d;a_i], \tag{4}$$

We also run a bidirectional LSTM RNNs as a question encoder which runs over the word embeddings of question, and concatenate the final state of forward RNNs and backward RNNs to obtain the representation of question $h^q = [\vec{h}_m^q;\overleftarrow{h}_1^q]$.

Assume that the latent variable z follow Gaussian distribution, the approximation network $Q_\phi(z|X,A) \sim N(\mu,\sigma^2\mathcal{I})$ and the prior network $P_\theta(z|X,A) \sim N(\mu',\sigma'^2\mathcal{I})$. When we obtain the passage representation h^D, the answer representation h^a and the question representation h^q, we can calculate the mean and variance of Q and P:

$$\begin{bmatrix} \mu \\ log(\sigma^2) \end{bmatrix} = W_q \begin{bmatrix} h^D \\ h^a \\ h^q \end{bmatrix} + b_q, \tag{5}$$

[3]We use the notation $[\cdot;\cdot]$ to denote concatenation of two vectors throughout the paper.

$$\begin{bmatrix} \mu' \\ log(\sigma'^2) \end{bmatrix} = W_{p2}(tanh(W_{p1} \begin{bmatrix} h^D \\ h^a \end{bmatrix} + b_{p2})) + b_{p1}, \quad (6)$$

where W_q, b_q, W_{p1}, b_{p1},W_{p2}, and b_{p2} are learning parameters. We then obtain samples of z from the approximation network Q (training) or the prior network P (testing) using the reparameterization trick [Kingma and Welling, 2013] and concatenate h^D, h^a, z and question type label c (one-hot representation). Finally, we initialize the hidden state of the decoder with the nonlinear transformation of these concatenated representation $s_0 = tanh(W_0[h^D; h^a; z; c] + b_0)$, where W_0 and b_0 are learning parameters.

At the stage of decoding, we employ an attention-based LSTM decoder to decode the source passage and answer information to generate questions. At decoding time step t, the LSTM decoder reads the previous word embedding y_{t-1} and context vector C_{t-1} to compute the new hidden state s_t. The context vector C_t for current time step t is computed through the attention mechanism [Luong *et al.*, 2015], which matches the current decoder state s_t with each hidden state h^d in the context encoder to get an importance score. The importance scores are then normalized to get the current context vector by weighted sum:

$$\begin{cases} s_t = LSTM(y_{t-1}, C_{t-1}, s_{t-1}) \\ e_{t,i} = v^T tanh(W_e s_{t-1} + U_e h_i^d) \\ \alpha_{t,i} = \dfrac{exp(e_{t,i})}{\sum_{i=1}^{n} exp(e_{t,i})} \\ C_t = \sum\limits_{i=1}^{n} \alpha_{t,i} h_i^d \end{cases} \quad (7)$$

where W_e, U_e and v^T are learning parameters. Finally, the probability of each target word y_t is predicted based on all the words that are generated previously (i.e., $y_{<t}$), and input text passage X.

$$P(y_t|X, y_{<t}) = softmax(V'(V[s_t; C_t] + b) + b') \quad (8)$$

where V', V, b' and b are learnable parameters .While training our generative model with LSTM decoder, the vanishing latent variable problem [Bowman *et al.*, 2016] make the training process difficult as the decoder tends to ignore the latent variable. We use similar techniques in [Bowman *et al.*, 2016] to overcome this problem by KL annealing and word drop decoding. The former gradually increases the weight of KL term from 0 to 1 over the course of model training and the latter uses a certain percentage of dropout for target words in the decoder.

3.2 Discriminative Model

In our question generation model, every generated sample has a corresponding class label c and the generator G use the passage X, the answer A, the latent variable z and question class label c to generate questions . Similar to the discriminator of AC-GAN [Odena *et al.*, 2017], our discriminator D has a binary classifier and an auxiliary classifier. The binary classifier aims at distinguishing the input question sequence whether is generated by humans or synthesized by machines and the auxiliary classifier predicts the probability of the question class. We firstly obtain the answer semantic representation and the question semantic representation using the similar way in generative model. And then the both representations are encoded into a vector representation using a LSTM RNNs encoder, which finally is fed to 2-class softmax function and 6-class softmax function, returning a probability distribution over the input question being a machine generated question or human-generated question $P(q|Y)$ and a probability distribution over the question class $P(C|Y)$, respectively.

3.3 Parameters Training

Our question generation model adopts the adversarial training process in GAN framework. The discriminator is regard as a reward function to further improve the generator iteratively by dynamically updating the discriminator. We firstly pretrain our generator by minimizing the loss J_{ml} in equation 3. Once we obtain more realistic and high-quality questions generated by generator G, the discriminator D is trained to minimize the following loss function:

$$\begin{aligned} \mathcal{L}_\psi^D = & -E_{Y\sim p_{data}[logP(q=real|Y)+logP(C=c|Y)]} \\ & -E_{Y\sim G_\theta[logP(q=fake|Y)+logP(C=c|Y)]} \end{aligned} \quad (9)$$

When the discriminator D is obtained and fixed, we are ready to update the generator G. The loss function of our generator G has two parts: the loss computed by policy gradient (denoted by J_{pg}) and the loss J_{ml} in equation 3 (contains the KL divergence and the cross-entropy loss). According to the policy gradient theorem [Sutton *et al.*, 1999], we compute the gradient of J_{pg} w.r.t. the parameters τ:

$$\begin{aligned} \nabla_\tau J_{pg} = & \frac{1}{T} \sum_{t=1}^{T} \sum_{y_t} [\alpha R^D((Y_{1:t-1}, I), y_t) \\ & + \beta R^C((Y_{1:t-1}, I), y_t)] \cdot \nabla_\tau (G_\tau(y_t|Y_{1:t-1}, I)) \\ = & \frac{1}{T} \sum_{t=1}^{T} E_{y_t \in G_\tau} [\alpha R^D((Y_{1:t-1}, I), y_t) \\ & + \beta R^C((Y_{1:t-1}, I), y_t) \cdot \nabla_\tau (G_\tau(y_t|Y_{1:t-1}, I))] \end{aligned} \quad (10)$$

where $R^D((Y_{1:t-1}, I), y_t)$ and $R^C((Y_{1:t-1}, I), y_t)$ are the action-value function. The former denotes the obtained reward from D which equals to the probability of the input question sequence being a human-generated question, and the latter is also a reward which equals to the probability of the real class of the generated question. I denotes the input content consisting of the input passage X, the answer A, the latent variable z and the question class variable c, and T is the length of the question sequence and $Y_{1:t}$ is the generated question up to time step t. α and β are the scaling factor to control the weight of two different reward function. Formally, the objective function of G is $J = J_{pg} + (2 - \alpha - \beta)J_{ml}$. We use α and β to balance the magnitude difference between J_{pg} and J_{ml}. In above equation, we use stochastic gradient descent to update the parameters of model.

4 Experiments

4.1 Dataset

We conduct our experiments on the *SQuAD* dataset [Rajpurkar *et al.*, 2016], which is used for machine reading comprehension and consists of more than 100,000 questions posed by crowd workers on 536 high-PageRank Wikipedia articles. We extract $\{passage, answer, question\}$ triples to construct the training, development and test sets. As the official test set is not publicly available, we randomly and nearly halve the development set to build the new development and test sets. We pre-process the dataset using Stanford CoreNLP to tokenize sentence. Then all passages are turned to lowercase and truncated into 400 tokens. After pre-processing, the extracted training, development and test sets contain 83,889, 5,168 and 5,000 triples respectively.

4.2 Training Details

We use the top 50,000 most frequent tokens for the source and target vocabulary. All other tokens outside the vocabulary list are replaced by the UNK symbol. We set the dimension of word embedding to 300 and use the *glove.840B.300d* pre-trained embeddings [Pennington *et al.*, 2014] for initialization. The LSTM hidden unit size is set to 300 and the number of layers of LSTMs is set to 1 in both the encoder and the decoder. We update the model parameters using stochastic gradient descent with mini-batch size of 64. The learning rate of generator G and discriminator D is set to 0.001, 0.0002, respectively. We clip the gradient when its norm exceeds 5. The scaling factors α and β are set to 0.6 and 0.5. The latent z space size is set to 200. During decoding, we do beam search with a beam size of 3.

4.3 Approaches

We conduct several experiments based on the following approaches, where **H&S-M** and **seq2seq** are the two baseline methods.

H&S-M: A manually constructed rule based system modified on the code released by [Heilman, 2011].

seq2seq: We implement a sequence-to-sequence model with attention mechanism [Bahdanau *et al.*, 2014] for question generation.

seq2seq + GAN: We employ sequence-to-sequence model with attention mechanism in GAN framework to further improve the performance by using adversarial learning method.

seq2seq + z: We employ an attentional sequence-to-sequence model and introduce the latent variable z to capture diversity in potential questions.

seq2seq + z + GAN: We implement an adversarial training process for an attentional sequence-to-sequence model with the latent variable z.

seq2seq + z +c: Based on the above "seq2seq + z" approach, we introduce an observed variable c of question class in this approach to learn disentangled representations from z and generate the specific types of questions.

seq2seq + z + c + GAN: We also employ "seq2seq + z + c" approach in GAN framework to push machines to generate questions that are indistinguishable from human-generated questions.

4.4 Evaluation Metrics

The proposed method is evaluated using the evaluation package released by [Chen *et al.*, 2015]. The package includes BLEU 1, BLEU 2, BLEU 3, BLEU 4, METEOR and ROUGE-L evaluation scripts. In our task, BLEU measures the average n-gram precision between generated and ground-truth questions. BLEU-n is BLEU score that calculates n-gram matches for counting co-occurrences. METEOR is a recall-oriented metric, which computes the similarity between generated and reference questions by considering synonyms, stemming and paraphrases. ROUGE metric is commonly used to evaluate n-grams recall of the generated summaries with ground-truth sentences as references. We report ROUGE-L (longest common subsequence) score in our experiment.

To evaluate the quality of questions generated by our "seq2seq + z + c" approach, we compare it against the human evaluation and the rule-based "H&S-M" baseline approach. We mainly consider whether the generated questions are natural, grammatically correct and fluent and whether the questions match the text passage and answer. We randomly sample 100 $[passage, question, answer]$ triples and four human raters will be asked to rate the triples on a 1–5 level, where 1 indicates the lowest level and 5 indicates the highest level.

4.5 Results and Analysis

Table 1 lists the performance of different models. A sampling is required from the latent space distribution for the models contained the latent variable z. In our evaluation, we calculate the average score from the three sampling results. Due to the unknown question type, for the models which contain the class variable c, e.g. "seq2seq + z + c" and "seq2seq + z + c + GAN", the scores presented in the table are calculated by choosing the best among all question types. The best result from each question type is the average score from sampling three times of the latent variable z. The results in the table shown that the proposed "seq2seq + z + c + GAN" approach achieves the best performance across all evaluation metrics.

In Table 1, we note that the performance of "seq2seq + z" approach is comparable with "seq2seq" approach. It implies that introducing the latent variable in seq2seq model don't bring the performance improvement but capture the diversity in potential questions. "seq2seq + z + c" approach achieves slightly better performance than "seq2seq" and "seq2seq + z" approaches. The potential reason for this may be that it is easier for "seq2seq + z +c" approach to generate good questions from the latent space distribution by specifying the corresponding question type for an answer. In addition, we also notice that all these approaches achieve notable performance improvement when employed in GAN architecture. That indicates that the adversarial learning method indeed further improves the performance.

We also evaluate the quality of questions generated by our "seq2seq + z + c" approach and the "H&S-M" approach with human judges. The human evaluation results also shown in Table 1. Our approach outperforms "H&S-M" approach by 1.8 score, which indicates the question generated by our "seq2seq + z + c" approach are more natural and more related to the given text passage and answer.

Model	BLEU1	BLEU2	BLEU3	BLEU4	METEOR	Rouge-L	HUMAN
H&s-m	37.2	21.5	14.32	9.54	14.35	28.88	2.3
seq2seq	42.3	24.9	16.7	12.21	16.24	39.18	-
seq2seq+GAN	43.42	25.96	17.5	12.98	17.21	40.23	-
seq2seq+z	43.1	24.71	16.54	12.4	16.43	38.96	-
seq2seq+z+GAN	44.27	25.68	17.52	13.21	17.38	40.31	-
seq2seq+z+c	43.3	25.1	16.82	12.56	16.97	39.3	-
seq2seq+z+c+GAN	**44.42**	**26.03**	**17.6**	**13.36**	**17.7**	**40.42**	**4.1**

Table 1: Evaluation results of different approaches on *SQuAD* test dataset from automatic evaluation metrics and human evaluation. The best performing method for each column is highlighted in bold.

I:	not all cells in a multicellular plant contain chloroplasts . all green parts of a plant contain chloroplasts the chloroplasts or more specifically the chlorophyll in them are what make the photosynthetic parts of a plant green .
Q:	what parts of plants have chloroplasts ?
G1:	what parts of plants contain chloroplasts ?
G2:	what parts of a plant 's cells are chlorophyll to be found ?
I:	from the late 1340s onwards people in the countryside suffered from frequent natural disasters such as droughts floods and the resulting famines and the government 's lack of effective policy led to a loss of popular support .
Q:	when did the yuan people suffer a series of natural disasters ?
G1:	when did people suffer from frequent natural disasters ?
G2:	when did people suffer from frequent natural disasters such as droughts ?
I:	throughout the 1980s and 1990s demand for a scottish parliament grew in part because the government of the united kingdom was controlled by the conservative party while scotland itself elected relatively few conservative mps .
Q:	whose control of the uk 's government helped fuel a desire for a scottish parliament ?
G1:	what political party controlled the government of the uk ?
G2:	who controlled the government of the united kingdom during the 1980s and 1990s ?

Table 2: Examples of generated questions, I is the input passage, Q is the gold question and G1 and G2 are the "seq2seq + z + GAN" generated questions by sampling twice from the latent space. The underlined words are the target answers.

I:	the plant cells which contain chloroplasts are usually parenchyma cells though chloroplasts can also be found in collenchyma tissue . a plant cell which contains chloroplasts is known as a chlorenchyma cell . a typical chlorenchyma cell of a land plant contains about 10 to 100 chloroplasts .
Q:	what plant cells have chloroplasts in them ?
G1:	who contains chloroplasts in plant ?
G2:	what parts of a plant contain chloroplasts ?
G3:	which parts of a plant contain chloroplasts ?
G4:	how many parts of a plant contain chloroplasts ?
G5:	when in plant are chloroplasts found ?
G6:	where are chloroplasts located in plant ?
I:	at the start of the war no french regular army troops were stationed in north america and few british troops . new france was defended by about 3,000 troupes de la marine companies of colonial regulars some of whom had significant woodland combat experience .
Q:	how much british military was in north america at start of war ?
G1:	who was stationed in north america at the beginning of the war?
G2:	what troops was stationed in north america ?
G3:	which troops were in north america
G4:	how many british troops were stationed in north america ?
G5:	when were the french troops troops stationed in north america ?
G6:	where were the french troops stationed in north america

Table 3: Examples of generated questions, I is the input passage, Q is the gold question and G1 $\sim$ G6 are the "seq2seq + z + c + GAN" generated questions by specifying the question type. The underlined words are the target answers.

We present question samples generated by our "seq2seq + z + GAN" approach and "seq2seq + z + c + GAN". The former aims to capture diversity in potential questions. Therefore, we sample twice from the latent space z, generating the two different expression of questions for one text passage and a given answer. The corresponding examples is shown in Table 2. It is observed that the generated questions in Table 2 indeed show diversity but tend to generate a single type of question. For instance, the first example shown in Table 2 is WHAT-type questions. However, it is easy to know that other question types such as WHICH-type and WHERE-type are also suitable for this example. It is hard to generate the different types of questions only by repeatedly sampling from the latent space. Therefore, we explore to generate specific types of questions using an observed variable c in "seq2seq + z + c + GAN" approach. We show some different types of questions generated by our method in Table 3. It is observed that it generates all types of questions for the given passage and answer although some types of questions don't match the given answer in the passage.

5 Conclusions and Future Work

In this paper, we proposed a natural question generation model employed in GAN framework, which uses the latent variable to capture the diversity and uses the observed variable to learn disentangled representation. Experimental results showed that our model could generate more readable and diverse questions with the specific types.

Except for question generation tasks, we hope to apply our model to dialogue system to generate utterances with different emotions. We plan to perform more work along this direction.

References

[Bahdanau *et al.*, 2014] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[Bowman *et al.*, 2016] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. Generating sentences from a continuous space. In *SIGNLL*, pages 10–21, 2016.

[Chali and Hasan, 2015] Yllias Chali and Sadid A. Hasan. Towards topic-to-question generation. *Computational Linguistics*, 41(1):1–20, 2015.

[Chen *et al.*, 2015] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO captions: Data collection and evaluation server. *CoRR*, 2015.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*, pages 1724–1734, 2014.

[Colby *et al.*, 1971] Kenneth Mark Colby, Sylvia Weber, and Franklin Dennis Hilf. Artificial paranoia. *Artif. Intell.*, 2(1):1–25, 1971.

[Du *et al.*, 2017] Xinya Du, Junru Shao, and Claire Cardie. Learning to ask: Neural question generation for reading comprehension. In *ACL*, pages 1342–1352, 2017.

[Goodfellow *et al.*, 2014] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[Graves, 2013] Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, 2013.

[Heilman and Smith, 2010] Michael Heilman and Noah A. Smith. Good question! statistical ranking for question generation. In *NAACL*, pages 609–617, 2010.

[Heilman, 2011] Michael Heilman. *Automatic factual question generation from text*. Carnegie Mellon University, 2011.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[Hu *et al.*, 2017] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. Controllable text generation. *CoRR*, abs/1703.00955, 2017.

[Kingma and Welling, 2013] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

[Li *et al.*, 2017a] Jiwei Li, Alexander H. Miller, Sumit Chopra, Marc'Aurelio Ranzato, and Jason Weston. Learning through dialogue interactions by asking questions. 2017.

[Li *et al.*, 2017b] Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. Adversarial learning for neural dialogue generation. In *EMNLP*, 2017.

[Luong *et al.*, 2015] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *EMNLP*, 2015.

[Odena *et al.*, 2017] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *ICML*, pages 2642–2651, 2017.

[Pennington *et al.*, 2014] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543, 2014.

[Rajpurkar *et al.*, 2016] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*, pages 2383–2392, 2016.

[Rezende *et al.*, 2014] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, pages 1278–1286, 2014.

[Schuster and Paliwal, 1997] Mike Schuster and Kuldip K. Paliwal. Bidirectional recurrent neural networks. *IEEE Trans. Signal Processing*, 45(11):2673–2681, 1997.

[See *et al.*, 2017] Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. In *ACL*, 2017.

[Serban *et al.*, 2016] Iulian Vlad Serban, Alberto García-Durán, Çaglar Gülçehre, Sungjin Ahn, Sarath Chandar, Aaron C. Courville, and Yoshua Bengio. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. In *ACL*, 2016.

[Sohn *et al.*, 2015] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *NIPS*, 2015.

[Sutton *et al.*, 1999] Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, 1999.

[Yu *et al.*, 2016] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. 2016.

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

[Zhao *et al.*, 2017] Tiancheng Zhao, Ran Zhao, and Maxine Eskénazi. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *ACL*, pages 654–664, 2017.

[Zhou *et al.*, 2017] Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. Neural question generation from text: A preliminary study. In *NLPCC*, pages 662–671, 2017.