PathQG: Neural Question Generation from Facts

Siyuan Wang[1], **Zhongyu Wei**[1,2]*, **Zhihao Fan**[1], **Zengfeng Huang**[1], **Weijian Sun**[3], **Qi Zhang**[4], **Xuanjing Huang**[4]
[1]School of Data Science, Fudan University, China
[2]Research Institute of Intelligent and Complex Systems, Fudan University, China
[3]Huawei Technologies Co., Ltd., China
[4]School of Computer Science, Fudan University, China
`{wangsy18,zywei,fanzh18,huangzf,qz,xjhuang}@fudan.edu.cn`
`sunweijian@huawei.com`

Abstract

Existing research for question generation encodes the input text as a sequence of tokens without explicitly modeling fact information. These models tend to generate irrelevant and uninformative questions. In this paper, we explore to incorporate facts in the text for question generation in a comprehensive way. We present a novel task of question generation given a query path in the knowledge graph constructed from the input text. We divide the task into two steps, namely, query representation learning and query-based question generation. We formulate query representation learning as a sequence labeling problem for identifying the involved facts to form a query and employ an RNN-based generator for question generation. We first train the two modules jointly in an end-to-end fashion, and further enforce the interaction between these two modules in a variational framework. We construct the experimental datasets on top of SQuAD and results show that our model outperforms other state-of-the-art approaches, and the performance margin is larger when target questions are complex. Human evaluation also proves that our model is able to generate relevant and informative questions.[1]

1 Introduction

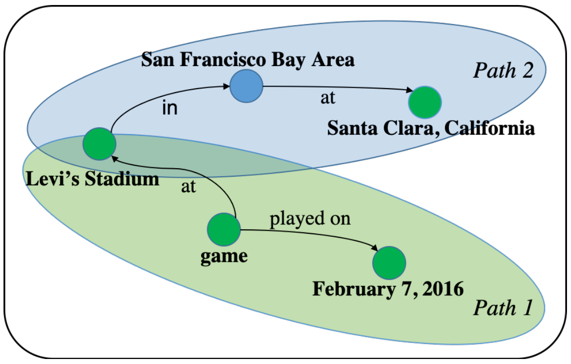
Question Generation (QG) from text aims to automatically construct questions from textual input (Heilman and Smith, 2010). It receives increasing attentions from research communities recently, due to its broad applications in scenarios of dialogue system and educational reading comprehension (Piwek et al., 2007; Duan et al., 2017). It can also help to augment the question set to enhance the performance of question answering systems.

Sentence:
The game was played on <u>February 7, 2016</u>, at Levi's Stadium in the San Francisco Bay Area at <u>Santa Clara, California</u>.

Generated questions:
- Q1: On what date was the game played ?
- Q2: Where was the Everton Fc Stadium located?

Ground truth questions:
- GTQ1: When was the game at Levi's Stadium played?
- GTQ2: Which state is the Levi's Stadium in?

(a) Machine generated questions (Qs) for an input text together with human generated ones (GTQs). Phrases underlined are the answers to ground-truth questions.



(b) Knowledge graph constructed based on the input text shown in top sub-figure. Two colored ellipsoid are two query paths related to two ground truth questions in sub-figure (a) respectively. Nodes in green are covered by ground-truth questions.

Figure 1: A sample paragraph from SQuAD with machine generated questions (Zhou et al., 2017) (a), ground truth questions (a) and corresponding knowledge graph (b).

Current QG systems mainly follow the sequence-to-sequence structure with an encoder for modeling the textual input and a decoder for text generation (Du et al., 2017). These neural-based models have shown promising performance, however, they suffer from generating irrelevant and uninformative questions. Figure 1a presents two sample questions generated by a nueral QG model. *Q2* contains irrelevant information "Everton Fc". Although *Q1* is correct, it is a safe play without mentioning any

*Corresponding author
[1]Our code is available at `https://github.com/WangsyGit/PathQG`.

specific information in the input text. One possible reason causing the problem is that current sequence-to-sequence models learn a latent representation for the input text without explicitly modeling semantic information included. We therefore argue that modeling facts in the input text can help to alleviate the problem of existing neural QG models.

Some researchers explore to incorporate the answer entity (Zhou et al., 2017) or a so called *question worthy phrase* (Wang et al., 2019) as the fact to guide the generation of target question and make some progresses. However, a complex question usually involves multiple facts. Therefore, a single word piece or phrase is not able to provide enough information for the generation. In this paper, we propose to represent facts in the input text as a knowledge graph (KG) and present a novel task of generating a question given a *query path* from the KG. More specifically, a KG contains a set of fact triples, and a *query path* is an ordered sequence of triples in the KG. A fact triple consists of two entities and their relationship.

Figure 1b shows the KG of the input text in Figure 1a and it includes two query paths. We can see that not all facts in a query path are mentioned in a specific target question (see Path 2 and GTQ2). Therefore, the model needs to extract the involved facts to form a query before it generates a question. Intuitively, we divide the task of question generation from a query path into two steps, namely, query representation learning and query-based question generation. We formulate the former step as a sequence labeling problem for identifying the involved facts to form a query. For query-based question generation, an RNN-based generator is used to generate the question word by word. We first train the two modules jointly in an end-to-end fashion (*PathQG* in Section 3). In order to further enforce the interaction between theses two modules, we employ a variational framework to train the two modules (Chen et al., 2018; Zhang et al., 2018) that treats query representation learning as an inference process from the query path taking the generated question as the target (*PathQG-V* in Section 4).

For model evaluation, we build the experimental environment on top of the benchmark dataset SQuAD (Rajpurkar et al., 2016). In specific, we automatically construct the KG for each piece of input text, and pair ground-truth questions with corresponding query paths from the KG. Experimental results show that our generation model outperforms other state-of-the-art QG models, especially when the questions are more complicated. Human evaluation also proves the effectiveness of our model in terms of both relevance and informativeness.

2 Task Definition

We first introduce some notations in our task:

- $x = (x_1, ..., x_n)$: an input text with n tokens, where x_i is the ith token;
- $\mathcal{G}$: a knowledge graph constructed from x, which is a set of fact triples $\{(e_1, r, e_2), ...\}$, where e_i is an entity and r_i is the relation between e_i and e_{i+1};
- $s = (e_1, r_1, e_2, ..., e_m)$: a query path in the knowledge graph, which is an ordered sequence of triples, and it's a subset of the $\mathcal{G}$;
- $y = (y_1, ..., y_{|y|})$: the generated question based on the x and s, where y_i is a token.

The task is described as following: given an input text x and its corresponding knowledge graph $\mathcal{G}$, our model aims to generate a question y_i based on a query path s_i from $\mathcal{G}$.

3 Path-based Question Generation

We divide the task of question generation from a given query path into two steps, namely, query representation learning and query-based question generation. A *Query Representation Learner* and a *Query-based Question Generator* are designed for the two steps separately. We directly combine these two modules into a unified framework *PathQG* and the overall architecture is illustrated in Figure 2.

3.1 Query Representation Learner

Query (representation) learner takes a query path s as input and learn the query representation L. Considering entities and relations in a query path have different contributions to generate the target question, we calculate their contribution weights for query representation learning.

3.1.1 Contribution Weight Calculation

We treat the task of contribution weight calculation as a sequence labeling task on the query path $s = (e_1, r_1, e_2, r_2, ..., e_m)$ taking entities and relations as tokens.

Context Encoding Considering the input text x can be useful to identify the weights of components in the path, we first encode the input text via a

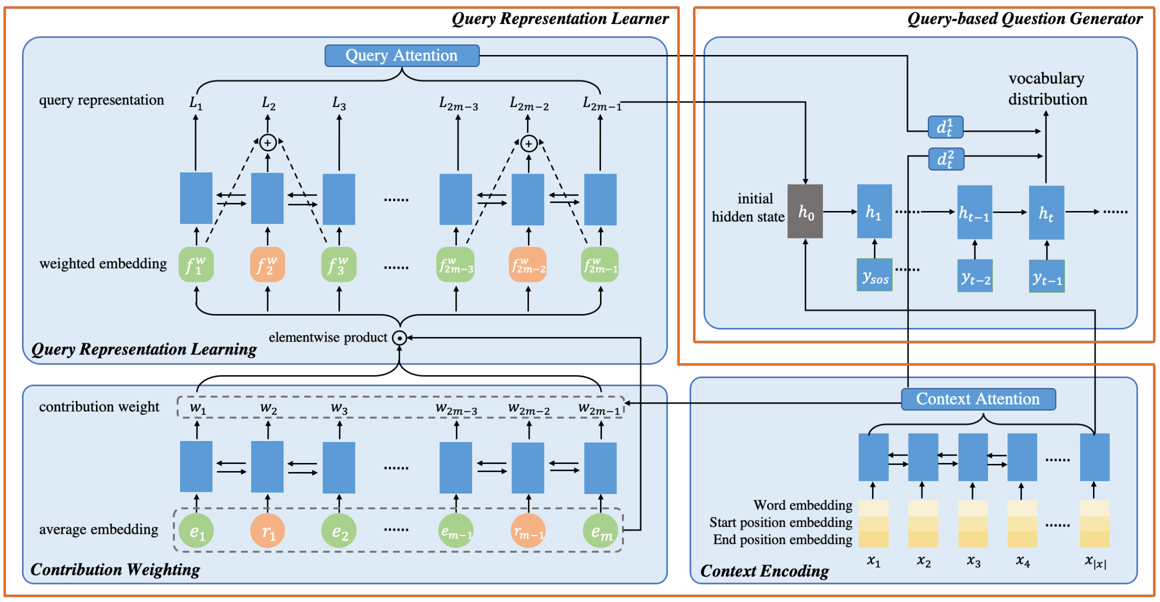


Figure 2: The overall architecture of *PathQG*.

context encoder. Following (Zhou et al., 2017), we use additional entity information (e.g., start, end entity of the query path) to improve the encoding of x. We use two BIO tags $b^a = (b^a_1, ..., b^a_n)$ and $b^e = (b^e_1, ..., b^e_n)$ to mark the positions of start and end entities in x. Then we concatenate the embeddings of x and two BIO tags as the input of the context encoder and use a bi-directional LSTM (Huang et al., 2015) to get the context states h^c_i as Eq. 1, where E_w and E_b are word embedding and tag embedding matrix respectively.

$$h^c_i = \text{BiLSTM}([E_w x_i;\ E_b b^a_i;\ E_b b^e_i], h^c_{i-1}) \quad (1)$$

Contribution Weighting Since each entity or relation in the path is also a sequence of tokens, we take the average pooling of their word embeddings as input $f_i = \text{average}(E_w s_i)$ where

$$s_i = \begin{cases} e_{(i+1)/2} & i \bmod 2 = 1 \\ r_{i/2} & i \bmod 2 = 0 \end{cases} \quad (2)$$

The encoding process of the path sequence labeling is as $h^s_i = \text{BiLSTM}(f_i, h^s_{i-1})$. And the encoding state h^s_i at each step i will attend to h^c and the attention output is computed as c_i. Then c_i will be concatenated with hidden state h^s_i to calculate the sigmoid probability of i_{th} component s_i in path as its contribution weight w_i where

$$w_i = P_{\theta_1}(s_i|x, s) = \sigma(\text{FFN}_2([h^s_i; c_i])) \quad (3)$$

where $\sigma(\cdot)$ is the sigmoid activation function and FFN_l is a l-layers feed-forward network.

3.1.2 Query Representation Learning

With the contribution weights of entities and their relations as $w = (w_1, w_2, ..., w_{2m-1})$, we encode the query path s in a weighted manner to learn the query representation. First we also utilize the average embeddings of entities and relations to compose the whole weighted query path as Eq. 4.

$$f^w = (w_1 \cdot f_1,\ w_2 \cdot f_2,\ ..., w_{2m-1} \cdot f_{2m-1}) \quad (4)$$

Considering a path has two different types of elements: entity and relation appearing in an alternating order, and the basic structural units constituting a path are triples, an RNN encoder is not able to capture these special structural information. Thus we adopt the recurrent skipping network (RSN) (Guo et al., 2019) instead of BiLSTM to encode the weighted path sequence and form a query representation as Eq. 5, 6,

$$\begin{aligned} \overrightarrow{h^L_i} &= \overrightarrow{\text{LSTM}}(f^w_i, \overrightarrow{h^L_{i-1}}) \\ \overleftarrow{h^L_i} &= \overleftarrow{\text{LSTM}}(f^w_i, \overleftarrow{h^L_{i+1}}),\ f^w_i \in f^w \end{aligned} \quad (5)$$

$$L_i = \begin{cases} [\overrightarrow{h^L_i}; \overleftarrow{h^L_i}] & i \bmod 2 = 1 \\ \text{FFN}_1([\overrightarrow{h^L_i}; f^w_{i-1}; \overleftarrow{h^L_i}; f^w_{i+1}]) & i \bmod 2 = 0 \end{cases} \quad (6)$$

where $L = (L_1, ..., L_{2m-1})$ is the learned query representation.

3.2 Query-based Question Generator

Taking the query representation L as input, we generate the corresponding question. Following the sequence-to-sequence paradigm of NQG model (Du et al., 2017), we take the concatenation of final query representation L_{2m-1} and final hidden state h_n^c of the input text from Eq. 1 as the initial state of the decoder and generate the question word by word. For the decoder, a LSTM is applied with attention mechanism and both sentence context and learned query L are utilized in the attention module.

The decoder attends to the learned query $L = (L_1, ..., L_{2m-1})$ and gets an attention-based query representation d_t^1. Besides, it also attends to the textual context states $h^c = (h_1^c, ..., h_n^c)$ and computes an attended context d_t^2. The h_t, d_t^1 and d_t^2 are concatenated to calculate the softmax probability distribution over the whole vocabulary,

$$P(y_t|y_{<t}, L, x) = \text{softmax}(\text{FFN}_2([h_t; d_t^1; d_t^2])) \tag{7}$$

where the y_t is the prediction at time t, and the generated question is $y = (y_1, ..., y_{|y|})$.

4 Variational Path-based Question Generation

The query representation learning can be treated as an inference process from the query path with the input text as the condition. Motivated by the variational models for KG reasoning, we propose a variational inference model *PathQG-V* to train the query learner and question generator to further enforce the interaction between them. Additionally, it introduces a posterior query learner to infer a posterior query distribution assuming the target question provided.

Compared with the original objective of *PathQG* as Eq. 8, the variational model aims to minimize its negative evidence lower bound (ELBO) as Eq. 9,

$$\log P(y|x, s) = \log \sum_L P_{\theta_1}(L|x, s) P_{\theta_2}(y|L, x) \tag{8}$$

$$\begin{aligned} -ELBO = KL(P_{\theta_3}(L|y, x, s) || P_{\theta_1}(L|x, s)) \\ - E_{P_{\theta_3}(L|y,x,s)}[\log P_{\theta_2}(y|L, x)] \end{aligned} \tag{9}$$

where $P_{\theta_1}(L|x, s)$, $P_{\theta_3}(L|y, x, s)$ and $P_{\theta_2}(y|L, x)$ are prior and posterior query distribution, and the likelihood of question y respectively. The structure of the variational model *PathQG-V* is shown in Figure 3. Note that the *prior query learner* and the *query-based question generator* are the same with query learner and question generator in Section 3.

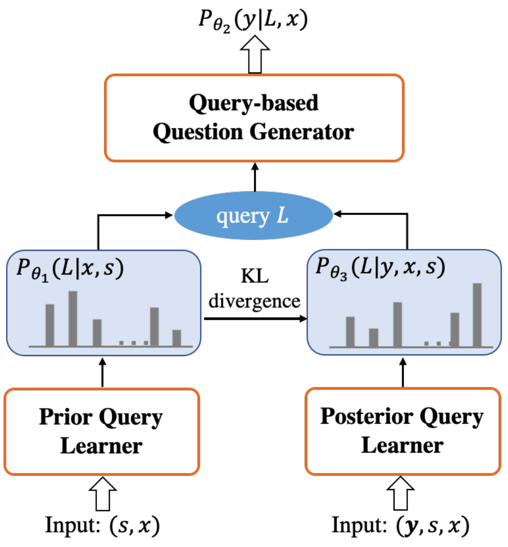


Figure 3: Overview of the *PathQG-V* model. x, s and y are the input text, query path and the question respectively.

4.1 Posterior Query Learner

The posterior query learner is designed in a similar manner as the query learner in Section 3.1, except that the target question is given. We incorporate the target question y in the same way as the input text x, where we employ a BiLSTM to encode the question y and get their hidden states $h^q = (h_1^q, ..., h_t^q)$. In the decoder of contribution weighting process, those question states are attended as same as context states h^c and get the attention output q_i at each step i. Then q_i together with c_i are concatenated with the encoding hidden state h_i^s to compute the posterior contribution weight of i_{th} component s_i in path as

$$w_i' = P_{\theta_3}(s_i|y, x, s) = \sigma(\text{FFN}_2([h_i^s; c_i; q_i])) \tag{10}$$

Then following Eq. 4, 5, 6, the posterior query representation $L' = (L_1', ..., L_{2m-1}')$ can be learned.

4.2 Optimization and Inference

During training period the posterior learned query representation L' is fed to the question generator, and the objective is to minimize the negative ELBO as Eq. 9. And the first term of negative ELBO can be viewed as Eq. 11:

$$\begin{aligned} \mathcal{L}_1 = KL(P_{\theta_3}((s_1, ..s_{2m-1})|y, x, s)|| \\ P_{\theta_1}((s_1, ..s_{2m-1})|x, s)) \end{aligned} \tag{11}$$

Then the $P_{\theta_2}(y|L,x)$ is the generation probability of question y and the log-likelihood can be rewritten as Eq. 12. We use the weighted path to form a query representation instead of sampling from the query distribution, therefore the second term of negative ELBO can be formulated as Eq. 13 where the expectation over posterior distribution $E_{P_{\theta_3}(L|y,x,s)}[\cdot]$ is omitted.

$$\log P_{\theta_2}(y|L,x) = \sum_{t=1}^{|y|} \log P_{\theta_2}(y_t|y_{<t},L,x) \tag{12}$$

$$\mathcal{L}_2 = -\log P_{\theta_2}(y|L,x,s) \tag{13}$$

To ensure the performance of query representation learner, we also add a contribution weighting loss defined as Eq. 14:

$$\mathcal{L}_3 = -\log P_{\theta_1}(L|x,s) \tag{14}$$

We combine all losses in a weighted manner as $\mathcal{L} = \lambda\mathcal{L}_1 + \mathcal{L}_2 + \beta\mathcal{L}_3$ to jointly train the framework, where λ and β are weighted hyper-parameters.

For the inference, only the prior query learner and the question generator are involved. The process is the same as PathQG.

5 Experiments

5.1 Experimental Dataset

Our experiments are conducted on SQuAD (Rajpurkar et al., 2016) consisting of 61,623 sentences. Each sentence is annotated with several questions together with their answers extracted from the input text. We build our experimental dataset on top of SQuAD. We construct knowledge graph for each sentence automatically and identify query paths for ground truth questions for evaluation. The resulted dataset consists of 89,976 tuples (input sentence x, query path s, ground truth question y).

KG construction We employ the scene graph parser (Schuster et al., 2015) for KG construction from a textual description. It identifies entities and their relationships from a text and build a scene graph. It turns out that the generated scene graph usually misses some key information in the text, thus we employ the part-of-speech tagger to extract verb phrases between entities to further enrich relationship labels. The extended scene graph is used as the knowledge graph for the input text. The average quantities of entities and facts in each KG are 6.53 and 4.68 respectively. The average information coverage rate of the input text by constructed KG is 68.52%. Note that our question generation models are compatible with KGs constructed by other methods.

Complex question set construction Our setting is motivated by the scenario where questions are related to multiple facts. We are then curious about the effectiveness of our model for complex question generation. Therefore we further construct a complex question set. A question is treated as complex if the corresponding query path contains more than 3 triples. The resulted complex question set contains 16,578 samples[2]. The detailed statistics of *complex* and *whole* datatset can be seen in Table 1. The datasets are split following Du et al. (2017).

dataset	train	valid	test	len. of ques.
complex	12,828	1,895	1,855	14.7
whole	68,704	10,313	10,959	13.3

Table 1: Statistics of complex and whole datasets. *len. of ques.* is the average number of tokens in questions.

Query path and question pairing We then identify corresponding query paths from the KG for ground truth questions. In practice, a path can be determined by a start node and an end node. We thus use answer entity of the question as the start node and use the entity identified in the question as the end node. If the question contains multiple entities, we take the one farthest to the start node in the KG as the end node. We ignore the edge directions to simplify the modeling of query path.

5.2 Implementation Details

We construct different vocabularies for input texts and questions respectively by keeping words which appear more than twice. Glove (Pennington et al., 2014) is used to initialize word embedding with dimension 300 and the embedding for BIO tag is randomly initialized of size 20. The size of hidden units in LSTM cell in all encoders is 300 while the size of the generation decoder is 1200. The hyper-parameters to balance weights of losses are chosen as $\lambda = 0.5$ and $\beta = 0.1$. We evaluate our model on validation set to choose parameters. During test

[2]The constructed KGs and complex question index can be downloaded from `https://www.disc.fudan.edu.cn/data/fudan_pathqg_data.zip`.

Model	BLEU 1	BLEU 2	BLEU 3	BLEU 4	METEOR	ROUGE$_L$	SPICE($\times$100)
NQG$_+$	49.89	32.84	23.54	17.25	18.90	43.17	31.19
AFPA	50.05	33.14	23.95	17.68	19.04	43.29	31.52
ASs2s	50.45	33.37	24.06	17.88	19.38	43.52	31.78
NQG$_+$(pl)	50.87	33.65	24.18	17.81	19.33	43.61	32.10
PathQG	51.15	34.14	24.60	18.24	19.59	43.60	32.21
PathQG-V	**52.46**	**34.91**	**25.19**	**18.48**	**20.04**	**43.79**	**32.41**
NQG$_{++}$	53.98	36.32	26.39	19.59	20.80	45.70	34.91

Table 2: Evaluation results on **complex** dataset. (**Bold**: best performance of each column.)

Model	BLEU 1	BLEU 2	BLEU 3	BLEU 4	METEOR	ROUGE$_L$	SPICE($\times$100)
NQG$_+$	48.66	31.23	21.71	15.44	18.23	43.17	28.30
AFPA	48.70	31.25	21.76	15.49	18.32	43.02	28.18
ASs2s	49.40	31.71	22.00	15.64	18.56	43.33	28.57
NQG$_+$(pl)	49.43	31.64	21.92	15.56	18.44	43.27	28.63
PathQG	49.45	31.95	22.22	15.79	18.56	43.44	28.69
PathQG-V	**50.14**	**32.25**	**22.48**	**15.98**	**18.85**	**43.46**	**28.88**
NQG$_{++}$	50.39	32.63	22.85	16.35	18.88	43.92	29.13

Table 3: Evaluation results on **whole** dataset. (**Bold**: best performance of each column.)

process, we use beam search of beam size 5. Refer to Appendices A for further information of training details and parameter numbers.

5.3 Models for Comparison

We compare our approach with some state-of-the-art models. For fair comparison, the start and the end nodes of the query path are provided for all models.

- **NQG$_+$** follows Zhou et al. (2017), which is an attention-based encoder-decoder model with the sentence as input and uses BIO tagging scheme to incorporate additional entity information (start and end nodes) to generate questions.
- **AFPA** (Sun et al., 2018) combines answer-focused model and position-aware model for question generation. For fair comparison, the model is re-trained with rich features including NE and POS removed and end entity provided.
- **ASs2s** (Kim et al., 2019) utilizes additional answer information via answer separation. For fair comparison, we do not implement the retrieval style word generator and the model is re-trained in our setting with the end entity supplied.
- **NQG$_+$(pl)** is an extension of **NQG$_+$**. Instead of learning a continuous latent query L, we sample entities and relations from the path via sequence labeling. Together with the start and end entities, those identified extra information are all encoded using BIO scheme for question generation.
- **PathQG** is our proposed generation framework consisting of a query representation learner and a query-based question generator. **PathQG-V** is the variational version of **PathQG** with an additional posterior query learner.
- **NQG$_{++}$** is an oracle model that is aware of all path information contained in the target question and encode them via BIO scheme. It can be treated as the upper bound of **NQG$_+$(pl)**. We present this result for reference.

5.4 Automatic Evaluation Results

For the automatic evaluation, we utlize some widely adopted metrics including BLEU 1-4 (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005) and ROUGE$_L$ (Lin and Hovy, 2003). Besides, we also compare results in the semantic content level by using a metric named SPICE (Anderson et al., 2016). It evaluates the similarity of scene graphs generated from candidate and reference questions. Evaluation results on both *whole* and *complex* datasets are shown in the Table 2 and 3. We have several findings:

- **PathQG-V** outperforms other models in terms of all metrics on both datasets by a considerable margin. This indicates the effectiveness of our variational inference framework for modeling the query path for better question generation.

- **PathQG** identifying involved entities and relations along the path performs better than **NQG$_+$**, **AFPA** and **ASs2s**, which demonstrates the effectiveness of introducing more related facts for question generation. And the improvement of **PathQG** compared to **NQG$_+$(pl)** shows the necessity of joint training.
- Our model generates larger improvement on the *complex* dataset compared to the *whole* dataset. This follows our intuition that questions related to longer query path are more complicated and our model has more advantage on these cases. Using length of query path to control the difficulty of questions is also a novel design (Gao et al., 2018).
- **NQG$_+$**, **AFPA** and **ASs2s** utilize answer for QG in different ways and our model **PathQG-V** follows the way of **NQG$_+$** for simplicity. From the improvement of **AFPA** and **ASs2s** compared to **NQG$_+$**, our model can further be adapted to follow them and performs better.
- Although good performance is achieved by **PathQG-V**, there is still a certain gap between it and the oracle model **NQG$_{++}$**. It shows the query learning from the path still has potential to be improved.

5.5 Human Evaluation Results

To better evaluate the quality of the generated questions, we conduct human evaluation through Amazon Mechanical Turk (AMT). We randomly choose 100 instances and 3 crowd annotators are invited to compare the questions generated by **PathQG-V** with **NQG$_+$**, **AFPA** and **ASs2s** in pair-wise . For each instance, the annotators are asked to read the text with the answer, and compare two candidate questions to determine which one is better in terms of three aspects respectively. (1) Fluency: the question is fluent. (2) Correctness: the question is consistent to the text and the answer. (3) Informativeness: the question contains specific information of the input text. The comparison results are shown in Figure 4. We can see that our model outperforms others in terms of all characteristics. This further proves that our model can generate more informative and consistent questions.

5.6 Further Analysis

In order to evaluate whether our model can utilize the facts in the input text to generate questions

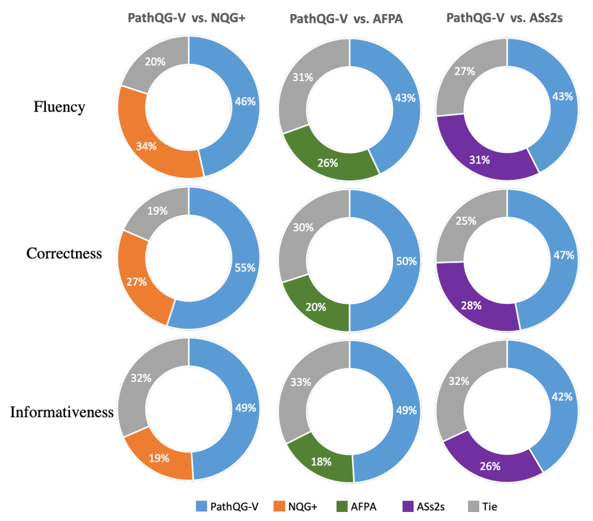


Figure 4: Pairwise comparison between the questions generated by **PathQG-V** and other methods in three characteristics. Each color is the percentage of annotators who consider the question generated by the corresponding method is better. "Tie" represents hard to tell.

with less irrelevant information. We analyse the relevance of the generated question to the text. We also demonstrate case studies.

Relevance of generated questions We evaluate the relevance of the generated questions to the input texts from different models by computing the overlapping rate. The results are presented in Figure 5. On both datasets, **PathQG-V** achieves the highest overlapping rate among all models, which shows our model can better utilize facts in the input text to generate more relevant questions. And the improvement of **PathQG** compared to other models reveals the effectiveness of learning involved entities and relations among path for question generation.

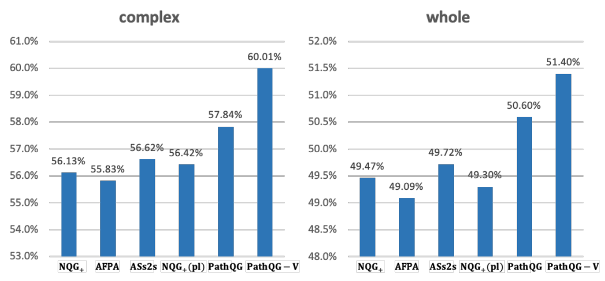


Figure 5: The overlapping rate between the input texts and the generated questions from different models.

Case study Two examples are presented in Figure 6. In sample 1, compared with the question generated by **NQG$_+$**, our generated question by

	Sample 1	Sample 2
Input	the three towns conurbation of plymouth , stonehouse and devonport enjoyed some prosperity during the late 18th and early 19th century and were enriched by a series of neo-classical urban developments designed by london architect john foulston .	because of its coastal location , the economy of plymouth has traditionally been maritime , in particular the defence sector with over 12,000 people employed and approximately 7,500 in the armed forces .
Path	(the three towns conurbation, of, plymouth, and, stonehouse, enjoyed, some prosperity, during, the late 18th, and, early 19th century, enriched by, a series, of, neo-classical urban developments, designed by, london architect john foulston).	(plymouth, of, the economy, been, the defence sector, with, over 12,000 people).
Question	**Reference**: what architect was noted for his neoclassical designs in plymouth ? **NQG+**: who designed the three towns in the 19th century ? **PathQG-V**: who designed the three towns from plymouth in the late 18th and early 19th century ?	**Reference**: how many plymouth residents are employed in defense ? **NQG+**: how many people work in the swazi economy ? **PathQG-V**: how many people work in the defence sector ?
Answer	john foulston.	12,000.

Figure 6: Two cases of input texts, paths, answers and questions generated by human, **NQG$_+$** and **PathQG-V**. Phrases underlined are irrelevant to the input text.

model **PathQG-V** is more informative and specific, which consists of information "plymouth" and "late 18th". In sample 2, our generated question is consistent to the input text while the one from **NQG$_+$** contains irrelevant phrase "swazi economy".

6 Related Work

Question Generation, aiming at generating questions from a range of inputs, such as raw text (Heilman and Smith, 2010), structured data (Serban et al., 2016) and images (Mostafazadeh et al., 2016; Fan et al., 2018a,b), has attracted increasing attention in recent years. Most previous studies on textual question generation are rule-based and transform a declarative sentence into an interrogative sentence according to hand-crafted patterns (Heilman and Smith, 2010; Heilman, 2011).

With the advance of neural network, Du et al. (2017) propose to apply a seq2seq structure with attention for automatic question generation. As follow-up, Zhou et al. (2017); Sun et al. (2018); Kim et al. (2019) propose to utilize the answers to decrease the generation uncertainty. Meanwhile, Song et al. (2018) and Li et al. (2019) explore to use answer-relevant context to guide question generation. Besides, some studies (Wang et al., 2017; Tang et al., 2017; Wang et al., 2019) take question generation as a subtask, and jointly learn it with other tasks, such as question answering and phrase extraction, which also help to alleviate the uncertainty and improve the generation performance.

Another stream of research for question generation is from KG to question. Reddy et al. (2017); Elsahar et al. (2018) explore to generate questions from a single KG triple using text as context information. It is close to our setting, but we are different in two aspects. First, we propose to form a query path consisting of multiple triples for question generation instead of a single triple. Second, the context we process is where the extracted triples from. This setting is more natural and different from using retrieved text as context as they did.

7 Conclusion and Future Work

In this paper, we propose to model facts in the input text as knowledge graph for question generation. We present a novel task of generating a question based on a query path from the constructed KG. We propose to learn query representation for question generation in a joint model and a variational inference model is also proposed. We extend the SQuAD dataset by automatic constructing KG for each input sentence and identifying corresponding query paths for ground truth questions. Experimental results proves the effectiveness of our proposed model qualitatively and quantitatively.

In the future, there can be two research directions. First, we would like to explore more explainable reasoning method for question generation, such as symbolic-based models. Second, novel evaluation metrics for question generation taking consistency and informativeness into consideration would be of interest.

Acknowledgments

This work is partically supported by National Natural Science Foundation of China (No. 71991471), Science and Technology Commission of Shanghai Municipality Grant (No.20dz1200600, No.18DZ1201000, 17JC1420200).

References

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. Spice: Semantic propositional image caption evaluation. In *European Conference on Computer Vision*, pages 382–398. Springer.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Wenhu Chen, Wenhan Xiong, Xifeng Yan, and William Wang. 2018. Variational knowledge graph reasoning. *Proceedings of the 2018 Conference of the NAACL.*

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*, pages 1342–1352.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. In *Proceedings of the 2017 Conference on EMNLP*, pages 866–874.

Hady Elsahar, Christophe Gravier, and Frederique Laforest. 2018. Zero-shot question generation from knowledge graphs for unseen predicates and entity types. *Proceedings of the 2018 Conference of the NAACL.*

Zhihao Fan, Zhongyu Wei, Piji Li, Yanyan Lan, and Xuanjing Huang. 2018a. A question type driven framework to diversify visual question generation. In *IJCAI*, pages 4048–4054.

Zhihao Fan, Zhongyu Wei, Siyuan Wang, Yang Liu, and Xuan-Jing Huang. 2018b. A reinforcement learning framework for natural question generation using bi-discriminators. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1763–1774.

Yifan Gao, Jianan Wang, Lidong Bing, Irwin King, and Michael R Lyu. 2018. Difficulty controllable question generation for reading comprehension. *Proceedings of the 2019 IJCAI Conference.*

Lingbing Guo, Zequn Sun, and Wei Hu. 2019. Learning to exploit long-term relational dependencies in knowledge graphs. *arXiv preprint arXiv:1905.04914.*

Michael Heilman. 2011. Automatic factual question generation from text. *Language Technologies Institute School of Computer Science Carnegie Mellon University*, 195.

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*, pages 609–617.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991.*

Yanghoon Kim, Hwanhee Lee, Joongbo Shin, and Kyomin Jung. 2019. Improving neural question generation using answer separation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6602–6609.

Jingjing Li, Yifan Gao, Lidong Bing, Irwin King, and Michael R Lyu. 2019. Improving question generation with to the point context. *arXiv preprint arXiv:1910.06036.*

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using N-gram cooccurrence statistics. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 71–78. Association for Computational Linguistics.

Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Margaret Mitchell, Xiaodong He, and Lucy Vanderwende. 2016. Generating natural questions about an image. *arXiv preprint arXiv:1603.06059.*

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on EMNLP*, pages 1532–1543.

Paul Piwek, Hugo Hernault, Helmut Prendinger, and Mitsuru Ishizuka. 2007. T2d: Generating dialogues between virtual agents automatically from text. In *International Workshop on Intelligent Virtual Agents*, pages 161–174. Springer.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Sathish Reddy, Dinesh Raghu, Mitesh M Khapra, and Sachindra Joshi. 2017. Generating natural language question-answer pairs from a knowledge graph using a rnn based question generation model. In *Proceedings of the 15th Conference of the EACL: Volume 1, Long Papers*, pages 376–385.

Sebastian Schuster, Ranjay Krishna, Angel Chang, Li Fei-Fei, and Christopher D Manning. 2015. Generating semantically precise scene graphs from textual descriptions for improved image retrieval. In *Proceedings of the fourth workshop on vision and language*, pages 70–80.

Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Linfeng Song, Zhiguo Wang, Wael Hamza, Yue Zhang, and Daniel Gildea. 2018. Leveraging context information for natural question generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 569–574.

Xingwu Sun, Jing Liu, Yajuan Lyu, Wei He, Yanjun Ma, and Shi Wang. 2018. Answer-focused and position-aware neural question generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3930–3939.

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

Siyuan Wang, Zhongyu Wei, Zhihao Fan, Yang Liu, and Xuanjing Huang. 2019. A multi-agent communication framework for question-worthy phrase extraction and question generation. *Association for the Advancement of Artificial Intelligence (AAAI)*.

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

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

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

A Appendices

A.1 Training Details

We train our proposed model and other comparison models on RTX 2080, using Adam optimizer with learning rate of 0.001, and decay at rate 0.96 per epoch, up to 20 epochs. Mini-batch of size 64 is taken for training. The dropout rate is 0.3 and we also clip the gradient once it exceeds 5. The average runtime for each method is listed in Table 4.

Model	runtime (minutes)
NQG$_+$	116
AFPA	160
ASs2s	97
NQG$_+$(pl)	127
PathQG	137
PathQG-V	170
NQG$_{++}$	113

Table 4: Average runtime of each method.

A.2 Model Parameters

We also compute the number of parameters in each model as shown in Table 5.

Model	number of parameters
NQG$_+$	23,769,132
AFPA	23,975,883
ASs2s	27,775,572
NQG$_+$(pl)	42,794,074
PathQG	33,775,934
PathQG-V	36,661,634
NQG$_{++}$	23,817,172

Table 5: Number of parameters in each model.