Reinforcement Learning Based Graph-to-Sequence Model for Natural Question Generation

Yu Chen*  
Rensselaer Polytechnic Institute  
cheny39@rpi.edu

Lingfei Wu*  
IBM Research  
lwu@email.wm.edu

Mohammed J. Zaki  
Rensselaer Polytechnic Institute  
zaki@cs.rpi.edu

Abstract

Natural question generation (QG) is a challenging yet rewarding task, that aims to generate questions given an input passage and a target answer. Previous works on QG, however, either (i) ignore the rich structure information hidden in the word sequence, (ii) fail to fully exploit the target answer, or (iii) solely rely on cross-entropy loss that leads to issues like exposure bias and evaluation discrepancy between training and testing. To address the above limitations, in this paper, we propose a reinforcement learning (RL) based graph-to-sequence (Graph2Seq) architecture for the QG task. Our model consists of a Graph2Seq generator where a novel bidirectional graph neural network (GNN) based encoder is applied to embed the input passage incorporating the answer information via a simple yet effective Deep Alignment Network, and an evaluator where a mixed objective function combining both cross-entropy loss and RL loss is designed for ensuring the generation of semantically and syntactically valid text. The proposed model is end-to-end trainable, and achieves new state-of-the-art scores and outperforms all previous methods by a great margin on the SQuAD benchmark.

1 Introduction

Natural question generation (QG) is a challenging yet rewarding problem. It has many useful applications including improving the question answering task (Chen et al., 2017, 2019a) by providing more training data (Du et al., 2017; Tang et al., 2017; Song et al., 2017; Yuan et al., 2017; Li et al., 2018), generating practice exercises and assessments for educational purposes (Heilman and Smith, 2010; Heilman, 2011; Danon and Last, 2017) and helping dialog systems, such as Alexa and Google Assistant, to kick-start and continue a conversation with human users (Mostafazadeh et al., 2016). While many other works focus on QG from images (Mostafazadeh et al., 2016; Fan et al., 2018; Li et al., 2018) or knowledge bases (Serban et al., 2016; Elsahar et al., 2018), in this work, we focus on QG from textual data.

Conventional methods (Mostow and Chen, 2009; Heilman and Smith, 2010; Heilman, 2011; Hussein et al., 2014) for QG rely on heuristic rules or hand-crafted templates, with low generalizability and scalability. Recent attempts have focused on neural network-based approaches that do not require manually-designed rules and are end-to-end trainable. Inspired by neural machine translation, these approaches formulate the QG task as a sequence-to-sequence (Seq2Seq) learning problem, applying various types of encoders and decoders, and have shown promising results (Du et al., 2017; Zhou et al., 2017; Song et al., 2018a, 2017; Kumar et al., 2018a). However, these methods ignore the hidden structural information associated with a word sequence such as the syntactic parsing tree. Thus, they may fail to utilize the rich text structure which complements the simple word sequence.

It has been observed that in general, training Seq2Seq models using cross-entropy based objectives has some limitations such as exposure bias and inconsistency between train/test measurement (Ranzato et al., 2015; Wu et al., 2016; Norouzi et al., 2016; Paulus et al., 2017; Gong et al., 2019), and thus does not always produce the best results on discrete evaluation metrics. To tackle these issues, some recent QG approaches (Song et al., 2017; Kumar et al., 2018b) aim at directly optimizing evaluation metrics using Reinforcement Learning (RL) (Williams, 1992). However, existing approaches generally do not consider joint mixed objective functions with both semantic and syntactic constraints for guiding text
passage.

Early works on neural QG did not take into account the target answer information when generating a question. Recent works (Zhou et al., 2017; Song et al., 2018a, 2017; Kumar et al., 2018a; Kim et al., 2018; Liu et al., 2019) have explored various ways of utilizing the target answer to guide the generation of the question, resulting in more relevant questions with better quality. These methods mainly focus on either simply marking the answer location in the passage (Zhou et al., 2017; Liu et al., 2019) using complex passage-answer matching strategies (Song et al., 2018a, 2017), or carefully separating answers from passages when applying a Seq2Seq model. However, they neglect potential semantic relations between word pairs and thus do not explicitly model the global interactions among sequence words in the embedding space.

To address the aforementioned issues, we propose a reinforcement learning based generator-evaluator architecture for QG. Our generator employs a modified graph-to-sequence (Graph2Seq) (Xu et al., 2018a) model that encodes the graph representation of a passage using a graph neural network (GNN) based encoder, and outputs a question sequence using a RNN decoder. Our evaluator is trained by optimizing a mixed objective function combining both cross-entropy loss and RL loss which is based on discrete evaluation metrics, where we first pretrain the model with regular cross-entropy loss.

We design a two-stage training strategy to align passage graphs and investigate their performance impact on a GNN encoder.

- The proposed model achieves new state-of-the-art scores and outperforms all previous methods by a great margin on the SQuAD (Rajpurkar et al., 2016) benchmark.

## 2 An RL-based Generator-Evaluator Architecture

Given a passage $X^p = \{x^p_1, x^p_2, \ldots, x^p_N\}$ and a target answer $X^a = \{x^a_1, x^a_2, \ldots, x^a_L\}$, the task of natural question generation is to generate the best natural language question $Y = \{y_1, y_2, \ldots, y_T\}$ which maximizes the conditional likelihood $Y = \arg\max Y P(Y | X^p, X^a)$.

We use 300-dim GloVe (Pennington et al., 2014) embeddings and 1024-dim BERT (Devlin et al., 2018) embeddings to embed each word in the passage and the answer. Let us denote $g^p_i$ and $g^a_j$ as the GloVe embeddings of passage word $x^p_i$ and answer word $x^a_j$, respectively. Similarly, their corresponding BERT embeddings are denoted by $BERT^p_i$ and $BERT^a_j$, respectively.

### 2.1 Deep Alignment Network

Answer information is crucial for generating relevant questions from a passage. Unlike previous methods that simply mark the answer location in the passage (Zhou et al., 2017), we propose the Deep Alignment Network (DAN) for incorporating answer information to the passage.

We highlight our contributions as follows:

- We propose a novel RL based Graph2Seq model for natural question generation.
- We design a simple yet effective Deep Alignment Network for explicitly modeling answer information.
- We design a novel bidirectionally Gated Graph Neural Network to process directed passage graphs.
- We design a two-stage training strategy to train the proposed model with both cross-entropy and RL losses.
- We explore different ways of constructing passage graphs and investigate their performance impact on a GNN encoder.

The proposed model achieves new state-of-the-art scores and outperforms all previous methods by a great margin on the SQuAD (Rajpurkar et al., 2016) benchmark.
In this work, we define the attention score $\beta_{i,j} \propto \exp(\text{ReLU}(WX^p_i)\text{T}\text{ReLU}(WX^a_j))$ where $W \in \mathbb{R}^{d \times d_a}$ is a trainable model parameter with $d$ being the hidden state size.

2.1.1 Word-level Alignment

We first do soft-alignment at the word-level based on the GloVe embeddings of the passage and the answer. Specifically, we compute the aligned answer embeddings by $f_{\text{align}}(X^p) = \text{Align}(g^p; g^a, g^o)$, where $f_{\text{align}}(x^p_i)$ is the aligned answer embedding for the $i$-th passage word.

For each passage word $x^p_i$, we concatenate its GloVe embedding $g^p_i$, BERT embedding $\text{BERT}^p_i$, linguistic feature (i.e., case, NER and POS) vector $f_{\text{feat}}(x^p_i)$ and the aligned answer embedding $f_{\text{align}}(x^p_i)$ to obtain a $d_p$-dim vector $w^p_i$. A bidirectional LSTM (Hochreiter and Schmidhuber, 1997) is applied to the passage embeddings $W^p = \{w^p_i\}_{i=1}^N$ to obtain contextualized passage embeddings $H^p \in \mathbb{R}^{d \times N}$. Similarly, for each answer word $x^a_i$, we concatenate its GloVe embedding $g^a_i$ with its BERT embedding $\text{BERT}^a_i$ to obtain a $d_a$-dim vector $w^a_i$. Another BiLSTM is applied to the answer embeddings $W^a = \{w^a_i\}_{i=1}^L$ to obtain contextualized answer embeddings $H^a \in \mathbb{R}^{d \times L}$.

2.1.2 Hidden-level Alignment

We proceed to do soft-alignment at the hidden state level based on the contextualized passage and answer embeddings. Similarly, we compute the aligned answer embeddings by $f_{\text{align}}^2(X^p) = \text{Align}(g^p; \text{BERT}^p; H^p), [g^a; \text{BERT}^a; H^a]$. $H^p$.

Finally, we apply another BiLSTM to the concatenation of the contextualized passage embeddings $H^p$ and the aligned answer embeddings $f_{\text{align}}^2(X^p)$ to get a $d$ by $N$ passage embedding matrix $X$.

2.2 Bidirectional Graph Encoder

While a Recurrent Neural Network (RNN) is good at modeling local interactions among consecutive passage words, a Graph Neural Network (GNN) can better utilize the rich text structure and the global interactions among sequence words, and thus further improve the representations. Therefore, we construct a passage graph $G$ which consists of each passage word as a node, and apply a GNN to the graph.

2.2.1 Passage Graph Construction

While it is straightforward to apply GNNs to graph structured data, it is not clear what is the best way of representing text as a graph. In this work, we explore both static and dynamic ways of constructing a graph for textual data.

Syntax-based static graph construction: We construct a directed unweighted passage graph based on dependency parsing. For each sentence in a passage, we first get its dependency parse tree. Then, we connect those neighboring dependency parse trees by connecting those nodes that are at a sentence boundary and next to each other.

Semantic-aware dynamic graph construction: We dynamically build a weighted graph to model semantic relationships among passage words. We make the process of building such a graph depend on not only the passage, but also on the answer. Specifically, we first apply a self-attention mechanism to the word-level passage embeddings $W^p$ to compute an attention matrix $A$, serving as a
weighted adjacency matrix for the passage graph, defined as \( A = \text{ReLU}(UW^p)^T \text{ReLU}(UW^p) \) where \( U \) is a \( d \times d_p \) trainable weight.

Considering that a fully connected passage graph is not only computationally expensive but also makes little sense for graph processing, we proceed to extract a sparse and directed graph where the \( K \) nearest neighbors (including itself) as well as the associated attention scores (i.e., the remaining attentions scores are masked off) for each node. Finally, we apply a softmax function to these selected adjacency matrix elements to get two normalized adjacency matrices, namely, \( A^{-1} \) and \( A^+ \), for incoming and outgoing directions, respectively.

\[
A^{-1} = \text{softmax}(\hat{A}) \\
A^+ = \text{softmax}(\hat{A}^T) \\
\hat{A} = \text{topk}(A)
\]

Note that the supervision signal is able to back-propagate through the KNN-style graph sparsification operation since the \( K \) nearest attention scores are kept and used to compute the weighted adjacency matrices.

### 2.2.2 Bidirectional Gated Graph Neural Networks

Unlike the previous Graph2Seq work (Xu et al., 2018a) that uses bidirectional GraphSAGE (Hamilton et al., 2017), we instead propose a novel Bidirectional Gated Graph Neural Network (BiGGNN) to process the directed passage graph. We extend the Gated Graph Sequence Neural Networks (Li et al., 2015) to handle directed graphs. Node embeddings are initialized to the representation of node \( v \) as \( h^n_v \). To compute the graph-level representation, we first apply a linear projection to the node embeddings, and then apply a maxpooling over all node embeddings to get a \( d \)-dim vector \( h_G \).

And for the semantic-aware dynamic graph, we compute a weighted average for aggregation where the weights come from the normalized adjacency matrices \( A^{-1} \) and \( A^+ \), defined as,

\[
h^k_{N^{-i}(v)} = \sum_{u \in N^{-i}(v)} a^{-1}_{i,u} h^{k-1}_u \\
h^k_{N^+(v)} = \sum_{u \in N^+(v)} a^+_{i,u} h^{k-1}_u
\]

While (Xu et al., 2018a) learn separate node embeddings for both directions independently, we choose to use a fusion function to fuse the information aggregated in two directions in every hop, which we find works better.

\[
h^k_N = \text{Fuse}(h^k_{N^{-i}(v)}, h^k_{N^+(v)})
\]

And the fusion function is designed as a gated sum of two information sources,

\[
\text{Fuse}(a, b) = z \ast a + (1 - z) \ast b \\
z = \sigma(W_z[a; b; a \ast b; a - b] + b_z)
\]

where \( \sigma \) is a sigmoid function and \( z \) is a gating vector.

Finally, a Gated Recurrent Unit (GRU) (Cho et al., 2014) is used to update the node embeddings by incorporating the aggregation information,

\[
h^k_v = \text{GRU}(h^{k-1}_v, h^k_N)
\]

After \( n \)-hops of GNN computation where \( n \) is a hyperparameter of GNN computation where \( n \) is a hyperparameter of

### 2.3 Decoder

On the decoder side, we follow the previous works (See et al., 2017; Song et al., 2017) to adopt an attention-based (Bahdanau et al., 2014; Luong et al., 2015) LSTM model with copy (Vinyals et al., 2015; Gu et al., 2016) and coverage mechanisms (Tu et al., 2016). The decoder takes the graph-level embedding \( h_G \) followed by two separate fully-connected layers as initial hidden states (i.e., \( c_0 \) and \( s_0 \)) and the node embeddings \( \{h^n_v, \forall v \in G\} \) as the attention memory, and generates the output sequence one word at a time.
The particular decoder used in this paper closely follows the work of (See et al., 2017) and we refer the readers to (See et al., 2017) for more details on the decoder. We now briefly describe the decoder. Basically, at each decoding step \( t \), an attention mechanism learns to attend to the most relevant words in the input sequence, and computes a content vector \( h_t^c \) based on the current decoding state \( s_t \), the current coverage vector \( c_t \) and the attention memory. In addition, the generation probability \( P_{\text{gen}} \in [0, 1] \) is calculated from the context vector \( h_t^c \), the decoder state \( s_t \) and the decoder input \( y_{t-1} \). Next, \( P_{\text{gen}} \) is used as a soft switch to choose between generating a word from the vocabulary, or copying a word from the input sequence. We dynamically maintain an extended vocabulary which is the union of the usual vocabulary and all words appearing in a batch of source examples (i.e., passages and answers). Finally, in order to encourage the decoder to utilize the diverse components of the input sequence, a coverage mechanism is applied. At each step, we maintain a coverage vector \( c_t \), which is the sum of attention distributions over all previous decoder time steps. A coverage loss is also computed to penalize repeatedly attending to the same locations of the input sequence.

### 2.4 Policy Based Reinforcement Learning

The most widely used way to train a sequence learning model is to optimize the log-likelihood of the ground-truth output sequence at each decoding step, which is known as the teacher forcing algorithm (Williams and Zipser, 1989).

However, it has been observed that optimizing such cross-entropy based training objectives for sequence learning does not always produce the best results on discrete evaluation metrics (Ranzato et al., 2015; Wu et al., 2016; Norouzi et al., 2016; Paulus et al., 2017; Gong et al., 2019). There are two main limitations of this method. First, a model has access to the ground-truth sequence up to the next token during training but does not have such supervision when testing, resulting in accumulated errors. This gap of model behavior between training and inference is called exposure bias (Ranzato et al., 2015). Second, there is an evaluation discrepancy between training and testing. A model is optimized with cross-entropy loss during training while evaluated with discrete evaluation metrics during testing.

To address the above issues, we present an evaluator where a mixed objective function combining both cross-entropy loss and RL loss is designed for ensuring the generation of semantically and syntactically valid text. We learn a policy that directly optimizes an evaluation metric using REINFORCE (Williams, 1992). In this work, we utilize the self-critical policy gradient training algorithm (Rennie et al., 2017), which is an efficient REINFORCE algorithm that, rather than estimating the reward signal, or how the reward signal should be normalized, utilizes the output of its own test-time inference algorithm to normalize the rewards it experiences.

For this training algorithm, at each training iteration, the model generates two output sequences: the sampled output \( Y^s \), produced by multinomial sampling, that is, each word \( y_t^s \) is sampled according to the likelihood \( P(y_t^s | X, y_{<t}) \) predicted by the generator, and the baseline output \( \hat{Y} \), obtained by greedy search, that is, maximizing the output probability distribution at each decoding step. We define \( r(\hat{Y}) \) as the reward of an output sequence \( \hat{Y} \), computed by comparing it to corresponding ground-truth sequence \( Y^* \) with some reward metrics. The loss function is defined as:

\[
L_{\text{rl}} = (r(\hat{Y}) - r(Y^*)) \sum_t \log P(y_t^s | X, y_{<t})
\]  

As we can see, if the sampled output has a higher reward than the baseline one, we maximize its likelihood, and vice versa.

**Evaluation metric as reward function:** We use one of our evaluation metrics, BLEU-4, as our reward function \( f_{\text{eval}} \), which lets us directly optimize the model towards the evaluation metrics.

**Semantic metric as reward function:** One drawback of some evaluation metrics like BLEU is that they do not measure meaning, but only reward systems for n-grams that have exact matches in the reference system. To make our reward function more effective and robust, we additionally use word movers distance (WMD) as a semantic reward function \( f_{\text{sem}} \). WMD is the state-of-the-art approach to measure the dissimilarity between two sentences based on word embeddings (Kusner et al., 2015). Following (Gong et al., 2019), we take the negative of the WMD distance between a generated sequence and the ground-truth sequence and divide it by the sequence length as its semantic score.

We define the final reward function as:

\[
r(Y) = f_{\text{eval}}(Y, Y^*) + \alpha f_{\text{sem}}(Y, Y^*)
\]  

where \( \alpha \) is a scalar.
2.5 Training and Testing

We train our model in two stages. In the first state, we train the model using regular cross-entropy loss, defined as,

\[ L_{lm} = -\sum_{i} \log P(y_i^* | X, y_{<i}) + \lambda \text{covloss}_t \]  

(8)

where \( y_i^* \) is the word at the \( t \)-th position of the ground-truth output sequence and covloss\(_t\) is the coverage loss defined as \( \sum_{i} \min(a^T_i, c^T_i) \), with \( a^T_i \) being the \( i \)-th element of the attention vector over the input sequence at time step \( t \). Note that the scheduled teacher forcing strategy (Bengio et al., 2015) is adopted to alleviate the exposure bias problem.

In the second stage, we fine-tune the model by optimizing a mixed objective function combining both cross-entropy loss and reinforcement loss, defined as,

\[ L = \gamma L_{rl} + (1 - \gamma) L_{lm} \]  

(9)

where \( \gamma \) is a scaling factor controlling the trade-off between cross-entropy and reinforcement losses. A similar mixed-objective learning function has been used by (Wu et al., 2016; Paulus et al., 2017) for machine translation and text summarization.

During the testing phase, we use beam search of width 5 to generate our final predictions.

3 Experiments

In this section, we conduct evaluation of our proposed model against state-of-the-art models. Following previous works, we use the SQuAD dataset (Rajpurkar et al., 2016) as our benchmark, described below. The implementation of the model will be publicly available soon.

3.1 Data and Metrics

SQuAD contains more than 100K questions posed by crowd workers on 536 Wikipedia articles. Since the test set of the original SQuAD is not publicly available, the accessible parts (90%) are used as the entire dataset in our experiments. For fair comparison, we evaluated our model on both data split-1 (Song et al., 2018a)\(^1\) that contains 75,500/17,934/11,805 examples and data split-2 (Zhou et al., 2017)\(^2\) that contains 86,635/8,965/8,964 examples.

Following previous works, we report and use BLEU-4 (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005) and ROUGE-L (Lin, 2004) as our evaluation metrics (BLEU-4 and METEOR were initially designed for evaluating machine translation systems and ROUGE-L was initially designed for evaluating text summarization systems).

3.2 Model Settings

We keep and fix the GloVe vectors for the most frequent 70,000 words in the training set. For BERT embeddings, we compute BERT embeddings on the fly for each word in text using a (trainable) weighted sum of all BERT layer outputs. The embedding sizes of case, POS and NER tags are set to 3, 12 and 8, respectively. The size of all hidden layers is set to 300. We apply a variational dropout (Kingma et al., 2015) rate of 0.4 after word embedding layers and 0.3 after RNN layers. We set the neighborhood size to 10 for dynamic graph construction. The number of GNN hops is set to 3. During training, in each epoch, we set the initial teacher forcing probability to 0.75 and exponentially increase it to \( 0.75 \times 0.9999^i \) where \( i \) is the training step. We set \( \alpha \) in the reward function to 0.1, \( \gamma \) in the mixed loss function to 0.99, and the coverage loss ratio \( \lambda \) to 0.4. We use Adam (Kingma and Ba, 2014) as the optimizer and the learning rate is set to 0.001 in the pretraining stage and 0.00001 in the fine-tuning stage. We reduce the learning rate by a factor of 0.5 if the validation BLEU-4 score stops improving for three epochs. We stop the training when no improvement is seen for 10 epochs. We clip the gradient at length 10. The batch size is set to 60 and 50 on data split-1 and split-2, respectively. All hyper-parameters are tuned on the development set.

3.3 Baseline methods

We compare our model with the following baseline methods: NQG++ (Zhou et al., 2017), MPQG+R (Song et al., 2017), ASs2s (Kim et al., 2018) and CGC-QG (Liu et al., 2019).

3.4 Evaluation Results and Case Study

Table 1 shows the experimental results comparing against all baseline methods. First of all, as we can see, our full model G2S\(_{std} + \)BERT+RL achieves the new state-of-the-art scores on both data splits and outperforms all previous methods by a great margin. Notably, some previous state-of-the-art
**Methods**

| Methods            | BLEU-4 | METEOR | ROUGE-L | BLEU-4 |
|--------------------|--------|--------|---------|--------|
| NQG++ (Zhou et al., 2017) | –      | –      | –       | 13.29  |
| MPQG+R (Song et al., 2017) | 13.98  | 18.77  | 42.72   | 13.91  |
| ASs2s (Kim et al., 2018) | 16.20  | 19.92  | 43.96   | 16.17  |
| CGC-QG (Liu et al., 2019) | –      | –      | –       | 17.55  |
| G2S$_{sta}$ +BERT+RL (our model) | **17.94** | **21.76** | **46.02** | **18.30** |

Table 1: Evaluation results of baseline methods and our model on the SQuAD test set. Higher scores indicate better performance. Note that we only report BLEU-4 on Split-2 since most of the baselines only report this result.

**Passage:** for the successful execution of a project, effective planning is essential.

**Gold:** what is essential for the successful execution of a project?

**Seq2Seq:** what type of planning is essential for the project?

**G2S$_{sta}$ w/o DAN:** what type of planning is essential for the successful execution of a project?

**G2S$_{sta}$:** what is essential for the successful execution of a project?

**G2S$_{sta}$ +BERT:** what is essential for the successful execution of a project?

**G2S$_{sta}$ +BERT+RL:** what is essential for the successful execution of a project?

**Passage:** the church operates three hundred sixty schools and institutions overseas.

**Gold:** how many schools and institutions does the church operate overseas?

**Seq2Seq:** how many schools does the church have?

**G2S$_{sta}$ w/o DAN:** how many schools does the church have?

**G2S$_{sta}$:** how many schools and institutions does the church have?

**G2S$_{sta}$ +BERT:** how many schools and institutions does the church have?

**G2S$_{sta}$ +BERT+RL:** how many schools and institutions does the church operate?

Table 2: Generated questions on SQuAD split-2 test set. Target answers are underlined.

In Table 2, we further show a few examples that illustrate the quality of generated text given a passage under different models. As we can see, incorporating answer information helps the model identify the answer type of the question to be generated, thus makes the generated questions more relevant and specific. Also, we find our Graph2Seq model can generate more complete and valid questions compared to the Seq2Seq baseline. Lastly, it shows that fine-tuning the model using REINFORCE can improve the quality of the generated questions.

### 3.5 Ablation Study of the Proposed Model

| Methods                  | Split-2 BLEU-4 |
|--------------------------|---------------|
| G2S$_{sta}$ +BERT+RL     | 18.30         |
| G2S$_{sta}$ +BERT-fixed+RL | 18.20   |
| G2S$_{sta}$ +BERT         | 18.02         |
| G2S$_{sta}$ +BERT-fixed   | 17.86         |
| G2S$_{sta}$ +RL           | 17.49         |
| G2S$_{sta}$              | 16.96         |
| G2S$_{dyn}$              | 16.81         |
| G2S$_{dyn}$ w/o feat.    | 16.51         |
| G2S$_{sta}$ w/o feat.    | 16.65         |
| G2S$_{dyn}$ w/o DAN       | 12.58         |
| G2S$_{sta}$ w/o DAN       | 12.62         |
| G2S$_{sta}$ w/ DAN-word only | 15.92 |
| G2S$_{sta}$ w/ DAN-hidden only | 16.07 |
| G2S$_{sta}$ w/ GGNN-forward | 16.53 |
| G2S$_{sta}$ w/ GGNN-backward | 16.75 |
| G2S$_{sta}$ w/o BIGGNN (Seq2Seq) | 16.14 |
| G2S$_{sta}$ w/o BIGGNN, w/ GCN | **14.47** |

Table 3: Ablation study on the SQuAD split-2 test set. Higher scores indicate better performance.
SQuAD split-2 test set. By turning off the Deep Alignment Network (DAN), the BLEU-4 score of G2S$_{sta}$ dramatically drops from 16.96% to 12.62%, which indicates the importance of answer information for QG and shows the effectiveness of DAN. We have a similar observation for the G2S$_{dyn}$ model. Further experiments demonstrate that both word-level (G2S$_{sta}$ w/ DAN-word only) and hidden-level (G2S$_{sta}$ w/ DAN-hidden only) answer alignments in DAN are helpful. We can see the advantages of Graph2Seq learning over Seq2Seq learning on this task by comparing the performance between G2S$_{sta}$ and Seq2Seq. In our experiments, we also observe that doing both forward and backward message passing in the GNN encoder is beneficial. Surprisingly, using GCN (Kipf and Welling, 2016) as the graph encoder (and converting the input graph to an undirected graph) harms the performance. In addition, fine-tuning the model using REINFORCE can further improve the model performance in all settings (i.e., w/ and w/o BERT), which shows the benefits of directly optimizing evaluation metrics. Besides, we find that the pretrained BERT embedding has a considerable impact on the performance and fine-tuning BERT embedding even further improves the performance, which demonstrates the power of large-scale pretrained language models. Incorporating common linguistic features (i.e., case, POS, NER) also helps the overall performance to some extent. Lastly, we find that syntax-based static graph construction slightly performs better than semantic-aware dynamic graph construction, even though the latter one seems to be more powerful (i.e., answer-aware and history-aware) and can be optimized towards the QG task in an end-to-end manner. Another big advantage of dynamic graph construction we can see is that it does not rely on domain knowledge to construct the graph. We leave how to better construct a dynamic graph for textual data in an end-to-end manner as future work. Another interesting direction is to explore effective ways of combining both the static and dynamic graphs.

4 Related Work

4.1 Natural Question Generation

Early works (Mostow and Chen, 2009; Heilman and Smith, 2010; Heilman, 2011; Hussein et al., 2014) for QG focused on rule-based approaches that rely on heuristic rules or hand-crafted templates, with low generalizability and scalability. Recent attempts have focused on Neural Network (NN) based approaches that do not require manually-designed rules and are end-to-end trainable. Existing NN based approaches (Du et al., 2017; Zhou et al., 2017; Song et al., 2018a; Kumar et al., 2018a) rely on the Seq2Seq model with attention, copy or coverage mechanisms. In addition, various ways (Zhou et al., 2017; Song et al., 2018a, 2017; Kim et al., 2018) have been proposed to utilize the target answer so as to guide the generation of the question. To address the limitations of cross-entropy based sequence learning, some approaches (Song et al., 2017; Kumar et al., 2018b) aim at directly optimizing evaluation metrics using REINFORCE.

4.2 Graph Neural Networks

Over the past few years, graph neural networks (GNNs) (Kipf and Welling, 2016; Gilmer et al., 2017; Hamilton et al., 2017; Li et al., 2015) have drawn increasing attention. Recently, GNNs have been applied to extend the widely used Seq2Seq architectures (Sutskever et al., 2014; Cho et al., 2014) to Graph2Seq architectures (Xu et al., 2018a,b,c; Song et al., 2018b). Very recently, researchers have explored methods to automatically construct a graph of visual objects (Norcliffe-Brown et al., 2018) or words (Liu et al., 2018; Chen et al., 2019b) when applying GNNs to non-graph structured data.

5 Conclusion

We proposed a novel reinforcement learning based Graph2Seq model for natural question generation, where the answer information is utilized by a simple yet effective Deep Alignment Network and a novel bidirectional GNN is proposed to process the directed passage graph. Our two-stage training strategy takes the benefits of both cross-entropy based and REINFORCE based training when training a sequence learning model. On the SQuAD dataset, our proposed model achieves the new state-of-the-art scores and outperforms all previous methods by a great margin. We also explore different ways of constructing graphs of textual data for graph neural networks. In the future, we would like to investigate more effective ways of automatically learning graph structures from free text.
References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

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.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In Advances in Neural Information Processing Systems, pages 1171–1179.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. arXiv preprint arXiv:1704.00051.

Yu Chen, Lingfei Wu, and Mohammed J Zaki. 2019a. Bidirectional attentive memory networks for question answering over knowledge bases. arXiv preprint arXiv:1903.02188.

Yu Chen, Lingfei Wu, and Mohammed J Zaki. 2019b. Graphflow: Exploiting conversation flow with graph neural networks for conversational machine comprehension. arXiv preprint arXiv:1908.00059.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In EMNLP, pages 1724–1734.

Guy Danon and Mark Last. 2017. A syntactic approach to domain-specific automatic question generation. arXiv preprint arXiv:1712.09827.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. arXiv preprint arXiv:1705.00106.

Hady Elsahar, Christophe Gravier, and Frederique Leforest. 2018. Zero-shot question generation from knowledge graphs for unseen predicates and entity types. arXiv preprint arXiv:1802.06842.

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

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pages 1263–1272. JMLR. org.

Hongyu Gong, Suma Bhat, Lingfei Wu, Jinjun Xiong, and Wen-mei Hwu. 2019. Reinforcement learning based text style transfer without parallel training corpus. arXiv preprint arXiv:1903.10671.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. arXiv preprint arXiv:1603.06393.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In Advances in Neural Information Processing Systems, pages 1024–1034.

Michael Heilman. 2011. Automatic factual question generation from text.

Michael Heilman and Noah A Smith. 2010. Good question! statistical ranking for question generation. In Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pages 609–617. Association for Computational Linguistics.

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

Hafedh Hussein, Mohammed Elmogy, and Shawkat Guirguis. 2014. Automatic english question generation system based on template driven scheme. International Journal of Computer Science Issues (IJCISI), 11(6):45.

Yanghoon Kim, Hwanhee Lee, Joongbo Shin, and Kyomin Jung. 2018. Improving neural question generation using answer separation. arXiv preprint arXiv:1809.02393.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Durk P Kingma, Tim Salimans, and Max Welling. 2015. Variational dropout and the local reparameterization trick. In Advances in Neural Information Processing Systems, pages 2575–2583.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.

Vishwajeet Kumar, Kireeti Boorla, Yogesh Meena, Ganesh Ramakrishnan, and Yuan-Fang Li. 2018a. Automating reading comprehension by generating question and answer pairs. In Pacific-Asia Conference on Knowledge Discovery and Data Mining, pages 335–348. Springer.
Vishwajeet Kumar, Ganesh Ramakrishnan, and Yuan-Fang Li. 2018b. A framework for automatic question generation from text using deep reinforcement learning. arXiv preprint arXiv:1808.04961.

Matt Kusner, Yu Sun, Nicholas Kolkkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In International Conference on Machine Learning, pages 957–966.

Yikang Li, Nan Duan, Bolei Zhou, Xiao Chu, Wanli Ouyang, Xiaogang Wang, and Ming Zhou. 2018. Visual question generation as dual task of visual question answering. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 6116–6124.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2015. Gated graph sequence neural networks. arXiv preprint arXiv:1511.05493.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. Text Summarization Branches Out.

Bang Liu, Mingjun Zhao, Di Niu, Kunfeng Lai, Yancheng He, Haojie Wei, and Yu Xu. 2019. Learning to generate questions by learning what not to generate. arXiv preprint arXiv:1902.10418.

Pengfei Liu, Shuaichen Chang, Xuanjing Huang, Jian Tang, and Jackie Chi Kit Cheung. 2018. Contextualized non-local neural networks for sequence learning. arXiv preprint arXiv:1811.08600.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025.

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.

Jack Mostow and Wei Chen. 2009. Generating instruction automatically for the reading strategy of self-questioning.

Will Norcliffe-Brown, Stathis Vafeias, and Sarah Parisot. 2018. Learning conditioned graph structures for interpretable visual question answering. In Advances in Neural Information Processing Systems, pages 8344–8353.

Mohammad Norouzi, Samy Bengio, Navdeep Jaitly, Mike Schuster, Yonghui Wu, Dale Schuurmans, et al. 2016. Reward augmented maximum likelihood for neural structured prediction. In Advances In Neural Information Processing Systems, pages 1723–1731.

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. Association for Computational Linguistics.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. arXiv preprint arXiv:1705.04304.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pages 1532–1543.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. arXiv preprint arXiv:1606.05250.

Marc’Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. arXiv preprint arXiv:1511.06732.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 7008–7024.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. arXiv preprint arXiv:1704.04368.

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. arXiv preprint arXiv:1603.06807.

Lin Feng Song, Zhiguo Wang, and Wael Hamza. 2017. A unified query-based generative model for question generation and question answering. arXiv preprint arXiv:1709.01058.

Lin Feng Song, Zhiguo Wang, Wael Hamza, Yue Zhang, and Daniel Gildea. 2018a. 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.

Lin Feng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018b. A graph-to-sequence model for amr-to-text generation. arXiv preprint arXiv:1805.02473.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Advances in neural information processing systems, pages 3104–3112.
Duyu Tang, Nan Duan, Tao Qin, Zhao Yan, and Ming Zhou. 2017. Question answering and question generation as dual tasks. arXiv preprint arXiv:1706.02027.

Zhaopeng Tu, Zhendong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. arXiv preprint arXiv:1601.04811.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In Advances in Neural Information Processing Systems, pages 2692–2700.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning, 8(3-4):229–256.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. Neural computation, 1(2):270–280.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google’s neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144.

Kun Xu, Lingfei Wu, Zhiguo Wang, and Vadim Sheinin. 2018a. Graph2seq: Graph to sequence learning with attention-based neural networks. arXiv preprint arXiv:1804.00823.

Kun Xu, Lingfei Wu, Zhiguo Wang, Mo Yu, Liwei Chen, and Vadim Sheinin. 2018b. Exploiting rich syntactic information for semantic parsing with graph-to-sequence model. arXiv preprint arXiv:1808.07624.

Kun Xu, Lingfei Wu, Zhiguo Wang, Mo Yu, Liwei Chen, and Vadim Sheinin. 2018c. Sql-to-text generation with graph-to-sequence model. arXiv preprint arXiv:1809.05255.

Xingdi Yuan, Tong Wang, Caglar Gulcehre, Alessandro Sordoni, Philip Bachman, Sandeep Subramanian, Saizheng Zhang, and Adam Trischler. 2017. Machine comprehension by text-to-text neural question generation. arXiv preprint arXiv:1705.02012.

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 Conference on Natural Language Processing and Chinese Computing, pages 662–671. Springer.