Toward Subgraph Guided Knowledge Graph Question Generation with Graph Neural Networks

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

Knowledge graph question generation (QG) aims to generate natural language questions from KG and target answers. Most previous works mainly focusing on the simple setting are to generate questions from a single KG triple. In this work, we focus on a more realistic setting, where we aim to generate questions from a KG subgraph and target answers. In addition, most of previous works built on either RNN-based or Transformer-based models to encode a KG sugraph, which totally discard the explicit structure information contained in a KG subgraph. To address this issue, we propose to apply a bidirectional Graph2Seq model to encode the KG subgraph. In addition, we enhance our RNN decoder with node-level copying mechanism to allow directly copying node attributes from the input graph to the output question. We also explore different ways of initializing node/edge embeddings and handling multi-relational graphs. Our model is end-to-end trainable and achieves new state-of-the-art scores, outperforming existing methods by a significant margin on the two benchmarks.

1 Introduction

Recently years have seen a surge of interests in Question Generation (QG) in machine learning and natural language processing. The goal of QG is to generate a natural language question for a given form of data such as text (Du et al., 2017; Song et al., 2018a), images (Li et al., 2018), table (Bao et al., 2018), and Knowledge Graph (Seyler et al., 2017; Kumar et al., 2019). In this paper, we focus on studying QG based on the knowledge graphs.

Knowledge Graphs (KG) have drawn a significant amount of research attention in recent years, partially due to its huge potential for an accessible, natural way of retrieving questions from KGs without need for learning complex query languages such as SPARQL. In order to train a large KB question answering system, large amount of question-answer pairs are often needed, which could be a severe bottleneck in reality. Developing effective approach to generate high-quality question-answer pairs from KGs could significantly address the limited data availability issue for KB-QA. Motivated by this observation, recent research efforts on KG-QG can be categorized into two classes.

The first line of research focuses on generating samples questions from a single triple (Serban et al., 2016; Elsahar et al., 2018; Reddy et al., 2017) in KGs. These models typically apply a sequence-to-sequence model with copy mechanism for translating either key word list or a triple into a natural question. However, although it is relatively easy to produce high-quality question-answer pairs, these generated data does not necessarily help solve complex multi-hops reasoning QA task. More recently, (Kumar et al., 2019) presented a difficulty controllable multi-hop question generation system from KGs, which directly work on a KG subgraph and thus they are able to generate more complex questions from these subgraphs using transformer-based models. However, they still considered the KB subgraphs as a set of triples and did not explicitly treat them as intrisinc graph structures and model them.

The task of KB-QG generation has three unique challenges. The first one is how to learn a good representation of a KB subgraph. A KB subgraph has complex underlying structures such as node attributes, and multi-relation edges. Each node and edge could be long strings that consists of multiple words. The second challenge is how to automatically learn a good mapping between a sub-graph and a natural language question. How to "copy" some unusual node or edge information that are related with generated questions. The third challenge is how to effectively leverage the answer informa-

*Corresponding author.

tion for question generation in KB.

In order to address the aforementioned challenges, we introduce for the first time the Graph2Seq architecture for the task of QG from KGs to address the second challenge. We extend the regular GNN encoder to make it able to process directed and multi-relational KG subgraphs to solve the first challenge. We explore two different ways of handling multi-relational graphs. In addition, we propose a simple but elegant way to leverage the context information from answer to effectively handle the third challenge. Our extensive experimental results have demonstrated the effectiveness of our proposed model on two benchmarks datasets. Compared to all existing state-of-the-art baselines, our model significantly outperforms them by a large margin on three different metrics.

We highlight our main contributes as follows:

- We propose the Graph2Seq model for subgraph guided question generation from KGs. The proposed Graph2Seq model employs a bidirectional graph embeddings and we exploit two different graph encoders in order to effectively cope with KB subgraphs with directed and multi-relation edges.
- We extend the RNN decoder with a novel copy mechanism that allows the entire node attribute to be borrowed from the input KG subgraph when generating the output question.
- We compare two different ways of initializing node/edge embeddings when applying a GNN encoder to process KG subgraphs. In addition, we study the impact of directionality (i.e., bidirectional vs. unidirectional) on GNN encoder.
- Our experimental results show that our proposed model improves the current state-of-the-art BLEU-4 score from 11.57 to 29.40 and from 25.99 to 59.59 on WebQuestions and PathQuestions, respectively.

2 Related Work

2.1 Natural Question Generation from Knowledge Graphs

Early works (Seyler et al., 2015; Song and Zhao, 2016; Seyler et al., 2017) for QG from KGs focused on template-based approaches that require significant manual effort, and have low generalizability and scalability. Recently, Seq2Seq (Sutskever et al., 2014; Cho et al., 2014) based neural architectures have been applied to this task without resort to manually-designed templates and are end-to-end trainable. However, these methods (Serban et al., 2016; Elsahar et al., 2018; Reddy et al., 2017) only focus on generating simple questions from a single triple as they typically employ an RNN-based encoder which cannot handle graph-structured data. Very recently, Kumar et al. (2019) presented a Transformer (Vaswani et al., 2017) based encoder-decoder model that allows to encode a KG subgraph and generate multi-hop questions. This is probably the first NN-based method that deals with QG from a KG subgraph instead of just a single triple. However, their method treats a KG subgraph as a set of triples which on the one hand, does not distinguish between entities and relations while modeling the graph, and on the other hand, does not utilize the explicit connections among triples.

Instead of using a Transformer-based encoder which might be incapable of utilizing the graph structure, in this work, we propose to employ a graph neural network to encode a KG subgraph in a more natural manner. To the best of our knowledge, we are the first to introduce the Graph2Seq architecture to this QG from KGs task.

2.2 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; Chen et al., 2019b) have attracted increasing attention since they extend traditional Deep Learning approaches to non-euclidean data such as graph-structured data. GNNs have many successful applications in computer vision (Norcliffe-Brown et al., 2018), natural language processing (Xu et al., 2018a,b,c; Chen et al., 2019c) and recommender systems (Ying et al., 2018). Because by design GNNs can easily model graph-structured data, recently, a number of works have extended the widely used Seq2Seq architectures (Sutskever et al., 2014; Cho et al., 2014) to Graph2Seq architectures for various tasks including machine translation (Bastings et al., 2017; Beck et al., 2018), semantic parsing (Xu et al., 2018b), and graph(e.g., AMR, SQL and KG)-to-text generation (Xu et al., 2018a,c; Song et al., 2018b; Marcheggiani and

Perez-Beltrachini, 2018; Distiawan et al., 2018; Vougiouklis et al., 2018).

3 Approach

3.1 Problem Formulation

The task of question generation (QG) aims to generate natural language questions based on a given form of data, such as KG or tables (Seyler et al., 2015; Lebret et al., 2016), text (Du et al., 2017; Song et al., 2018a; Chen et al., 2020), or images (Li et al., 2018), where the generated questions need to be answerable from the input data. In this paper, we focus on QG from a KG subgraph, along with potentially target answers.

We assume that a KG subgraph is a collection of triples (i.e., subject-predicate-object), that can also be represented as a graph $\mathcal{G} = (V, E)$, where $V \in \mathcal{V}$ denotes a set of entities (i.e., subjects or objects) and $E \in \mathcal{E}$ denotes all the predicates connecting these entities. We denote by $\mathcal{V}$ and $\mathcal{E}$ the complete entity set and predicate set of the KG, respectively. We also assume that all the answers from the target answer set V^a are from the entity set V, which is the normal setting of the task of KBQA (Chen et al., 2019a). The task of QG from KGs is to generate the best natural language question consisting of a sequence of word tokens $\hat{Y} = \{y_1, y_2, ..., y_T\}$ which maximizes the conditional likelihood $\hat{Y} = \arg\max_Y P(Y|\mathcal{G}, V^a)$. Here T is the length of the question.

3.2 Encoding Layer

Let us denote V as a set of nodes (i.e., entities) $\{v_1, v_2, ..., v_n\}$ in a KG subgraph $\mathcal{G}$, where each node is associated with some attributes such as textual name or ID. Similarly, let us denote E as a set of edges (i.e., predicates) $\{e_1, e_2, ..., e_m\}$ in $\mathcal{G}$, where each edge is associated with some attributes such as textual name or ID.

3.2.1 Encoding Nodes and Edges

Before applying the GNN encoder to process a KG subgraph, we need to map nodes and edges to some initial embedding space that encodes their attributes. There are two common ways of encoding nodes and edges in a KG. One solution is based on global KG embeddings that are pretrained on the whole KG by some knowledge-base representation learning algorithms such as TransE (Bordes et al., 2013), while the other one is based on pretrained embeddings (e.g., GloVe (Pennington et al., 2014)) of words making up the textual attributes. In this work, we choose to encode nodes and edges based on word embeddings of their textual attributes in our main model. We suspect that it is relatively easier for a model to learn the mapping from the input KG subgraph to the output natural language question with both sides based on word embeddings. We will empirically compare and analyze the two encoding strategies in experiments.

Hence, in order to encode the nodes and edges in the KG subgraph, we apply two bidirectional LSTMs (Hochreiter and Schmidhuber, 1997) (i.e., one for nodes, and one for edges) to encode their associated textual names. And the concatenation of the last forward and backward hidden states of BiLSTM is used as the initial embeddings for nodes as well as edges.

3.2.2 Utilizing Target Answers

In the setting of KBQA (Chen et al., 2019a; Haussmann et al., 2019), it is usually assumed that the answers to a question are entities in some KG subgraph. As a dual task of KBQA, in this QG work, we assume that utilizing the answer information along with with the given KG subgraph could help generate more relevant questions. To this end, we apply a simple yet effective way of utilizing the answer information, where we introduce an additional learnable markup vector associated to each node/edge to indicate whether it is an answer or not.

Hence, the initial vector representation of a node/edge will be the concatenation of the BiLSTM output and the answer markup vector. We denote $\mathbf{X}^e = \{\mathbf{x}_1^e, \mathbf{x}_2^e, ..., \mathbf{x}_n^e\}$ and $\mathbf{X}^p = \{\mathbf{x}_1^p, \mathbf{x}_2^p, ..., \mathbf{x}_m^p\}$ as the vector representations of the entity nodes and predicate edges in a KG subgraph, respectively. And we keep $\mathbf{X}^e$ and $\mathbf{X}^p$ have the same embedding dimension d.

3.3 Bidirectional Graph-to-Sequence Generator with Copying Mechanism

While RNNs are good at modeling sequential data, by natural they cannot handle graph-structured data. One might need to linearize a graph to a sequence so as to apply an RNN-based encoder, which will definitely loss the rich structure information in the graph. Even though a Transformer-based encoder might be able to learn the semantic relations among the triples through the all-to-all attention, the explicit graph structure is totally discarded. In this work, we introduce a novel Graph2Seq model for

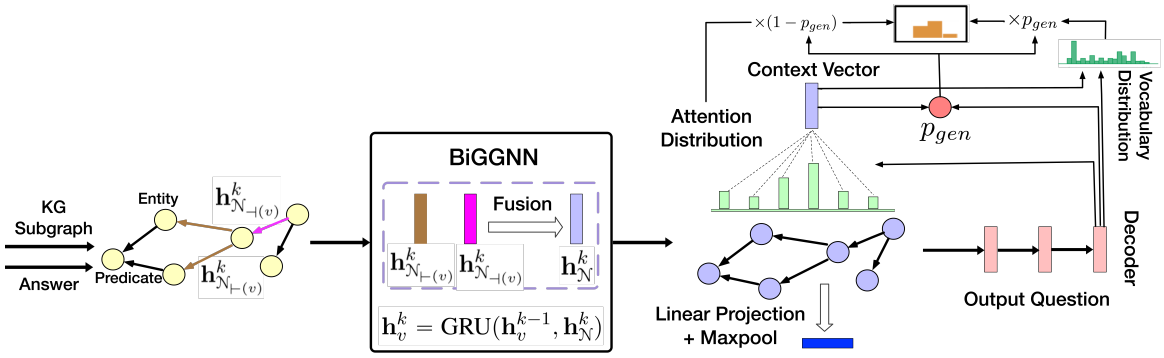

Figure 1: Overall architecture of the proposed model. Best viewed in color.

generating natural language questions from a KG subgraph.

3.3.1 Bidirectional Graph Encoder

Many existing GNNs (Kipf and Welling, 2016; Hamilton et al., 2017; Veličković et al., 2017) were not designed to process directed graphs such as a KG. Even though some GNN variants such as GGSNN (Li et al., 2015) and MPNN (Gilmer et al., 2017) are able to handle directed graphs via message passing across graphs, they do not model the bidirectional information when aggregating information from neighboring nodes for each node. As a result, messages can only be passed across graphs in a unidirectional way.

In this work, we introduce the Bidirectional Gated Graph Neural Network (BiGGNN) which extends GGSNN by learning node embeddings from both incoming and outgoing directions in an interleaved fashion when processing a directed graph. Similar bidirectional idea has been exploited in Xu et al. (2018a), which extended another popular variant of GNNs - GraphSAGE (Hamilton et al., 2017). While their method simply learns the node embeddings of each direction independently and concatenates them in the final step, BiGGNN fuses the intermediate node embeddings from both directions at every iteration.

The embedding $\mathbf{h}_v^0$ for node v is initialized to $\mathbf{x}_v$ that is a concatenation of the BiLSTM output and the answer markup vector. Same as GGSNN, BiGGNN performs message passing across graphs for a fixed number of hops, with the same set of network parameters shared at each hop. At each hop of computation, for every node in the graph, we apply an aggregation function that takes as input a set of incoming (or outgoing) neighboring node vectors and outputs a backward (or forward) aggregation vector. In principle, many order-invariant operators such as max or attention (Veličković et al., 2017) could be employed to aggregate neighborhood information. In this work, we use a simple average aggregator as below,

$$\begin{aligned}\mathbf{h}^k_{\mathcal{N}_{\dashv(v)}} &= \text{AVG}(\{\mathbf{h}_v^{k-1}\} \cup \{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}_{\dashv(v)}\}) \\ \mathbf{h}^k_{\mathcal{N}_{\vdash(v)}} &= \text{AVG}(\{\mathbf{h}_v^{k-1}\} \cup \{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}_{\vdash(v)}\})\end{aligned} \tag{1}$$

We then fuse the node embeddings aggregated from both directions at every hop.

$$\mathbf{h}^k_{\mathcal{N}_{(v)}} = \text{Fuse}(\mathbf{h}^k_{\mathcal{N}_{\dashv(v)}}, \mathbf{h}^k_{\mathcal{N}_{\vdash(v)}}) \tag{2}$$

The fusion function is computed as a gated sum of two information sources,

$$\begin{aligned}\text{Fuse}(\mathbf{a}, \mathbf{b}) &= \mathbf{z} \odot \mathbf{a} + (1 - \mathbf{z}) \odot \mathbf{b} \\ \mathbf{z} &= \sigma(\mathbf{W}_z[\mathbf{a}; \mathbf{b}; \mathbf{a} \odot \mathbf{b}; \mathbf{a} - \mathbf{b}] + \mathbf{b}_z)\end{aligned} \tag{3}$$

where $\odot$ is the component-wise multiplication, σ is a sigmoid function, and $\mathbf{z}$ 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.

$$\mathbf{h}_v^k = \text{GRU}(\mathbf{h}_v^{k-1}, \mathbf{h}^k_{\mathcal{N}_{(v)}}) \tag{4}$$

After n hops of GNN computation, where n is a hyperparameter, we obtain the final state embedding $\mathbf{h}_v^n$ for node v.

To compute the graph-level embedding, we first apply a linear projection to the node embeddings, and then apply max-pooling over all node embeddings to get a d-dim vector $\mathbf{h}^{\mathcal{G}}$.

3.3.2 Handling Multi-relational Graphs

Knowledge graphs are typically heterogeneous networks that contain a large number of edge types. However, many existing GNNs (Kipf and Welling, 2016; Hamilton et al., 2017; Li et al., 2015;

Veličković et al., 2017) are not directly applicable to such multi-relational graphs.

In order to model both the node and edge information with GNNs, researchers have extended GNNs by either having separate learnable weight matrices for different edge types or having explicit edge embeddings when performing message passing across graphs (Gilmer et al., 2017; Simonovsky and Komodakis, 2017). While the former solution is supposed to have severe scalability issues when handling graphs with a large number of edge types, the later one requires major modifications to existing GNN architectures. In this work, we explore two solutions to adapt GNNs to multi-relational graphs where one is based on Levi graph transformation (Levi, 1942), and the other is to explicitly introduce edge embeddings into the GNN architecture.

Levi graph transformation By converting a multi-relational KG subgraph to a Levi graph (Levi, 1942), we can directly apply regular GNNs without modification. Specifically, we treat all edges in the original graph as new nodes and add new edges connecting original nodes and new nodes, which results in a bipartite graph. For instance, in a KG subgraph, a triple "Mario_Siciliano place_of_birth Rome" with the entities "Mario_Siciliano" and "Rome" being nodes and the predicate "place_of_birth" being an edge, will be converted to "Mario_Siciliano → place_of_birth → Rome" where "place_of_birth" becomes a new node, and → indicates a new edge connecting an entity and a predicate. Note that since most KG subgraphs are sparse, the number of newly added nodes (and edges as well) will almost be linear to the number of original nodes.

Gated message passing with edge information We explore extending BiGGNN to explicitly incorporate edge embeddings when conducting message passing. We name the resultant variant as BiGGNN$_{edge}$. Specifically, we rewrite the node aggregation function Eq. (1) as follows,

$$\begin{aligned}\mathbf{h}^k_{\mathcal{N}_{\dashv(v)}} &= \text{AVG}(\{\mathbf{h}^{k-1}_v\} \cup \{f([\mathbf{h}^{k-1}_u; \mathbf{e}_{uv}]), \forall u \in \mathcal{N}_{\dashv(v)}\}) \\ \mathbf{h}^k_{\mathcal{N}_{\vdash(v)}} &= \text{AVG}(\{\mathbf{h}^{k-1}_v\} \cup \{f([\mathbf{h}^{k-1}_u; \mathbf{e}_{uv}], \forall u \in \mathcal{N}_{\vdash(v)}\})\end{aligned} \tag{5}$$

where f is a nonlinear function (i.e., linear projection + ReLU (Nair and Hinton, 2010)) applied to the concatenation of $\mathbf{h}^{k-1}_u$ and $\mathbf{e}_{uv}$, and $\mathbf{e}_{uv}$ is the embedding of the edge connecting node u and v.

3.3.3 RNN Decoder with Node-level Copying Mechanism

We adopt an attention-based (Bahdanau et al., 2014; Luong et al., 2015) LSTM decoder that generates the output sequence one word at a time. The decoder takes the graph-level embedding $\mathbf{h}^{\mathcal{G}}$ followed by two separate fully-connected layers as initial hidden states (i.e., $\mathbf{c}_0$ and $\mathbf{s}_0$) and the node embeddings $\{\mathbf{h}^n_v, \forall v \in \mathcal{G}\}$ as the attention memory. The particular attention mechanism used in our decoder closely follows See et al. (2017). Basically, at each decoding step t, an attention mechanism learns to attend to the most relevant nodes in the input graph, and computes a context vector $\mathbf{h}^*_t$ based on the current decoding state $\mathbf{s}_t$, the current coverage vector $\mathbf{c}^t$ and the attention memory.

We hypothesize that when generating natural language questions from a KG subgraph, it is very likely to directly mention (i.e., copy) entity names that are from the input KG subgraph even without rephrasing them. When augmented with copying mechanism (Vinyals et al., 2015; Gu et al., 2016), most RNN-based decoders are typically allowed to copy tokens from the input sequence. We instead extend the regular word-level copying mechanism to the node-level copying mechanism that allows copying node attributes (i.e., node names) from the input graph. At each step of decoding, the generation probability $p_{\text{gen}} \in [0, 1]$ is calculated from the context vector $\mathbf{h}^*_t$, the decoder state $\mathbf{s}_t$ and the decoder input y_{t-1}. Next, p_{gen} is used as a soft switch to choose between generating a word from the vocabulary, or copying a node attribute from the input graph.

3.4 Training and Testing

Following prior works on training sequential models, we minimize the cross-entropy loss, defined as,

$$\mathcal{L}_{lm} = \sum_t -\log P(y^*_t | X, y^*_{<t}) \tag{6}$$

where y^*_t is the word at the t-th position of the ground-truth output sequence. Scheduled teacher forcing (Bengio et al., 2015) is adopted to alleviate the exposure bias problem. During the testing phase, beam search is applied to generate the output.

Besides training our proposed model with the regular cross-entropy loss, we also explore minimizing a hybrid objective combining both the cross-entropy loss and Reinforcement Learning

(RL) (Williams, 1992) loss that is defined based on evaluation metrics. For more details on training the model with the hybrid objective function, please refer to Appendix A. We will evaluate this training strategy in our experiments.

4 Experiments

In this section, we conduct extensive experiments to evaluate the effectiveness of our proposed model. Besides, we want to examine whether the introduced GNN-based encoder works better than an RNN-based or Transformer-based encoder when encoding a KG subgraph for the QG task. In addition, we explore and analyze two different ways of handling multi-relational graphs with GNNs. Moreover, we empirically compare two different ways of initializing node and edge embeddings before feeding them into a GNN-based encoder. An experimental comparison between bidirectional GNN-based encoder and unidirectional GNN-based encoder is also provided. For model settings, please refer to Appendix B. The implementation of the model will be made publicly available upon the acceptance of this paper.

4.1 Baseline Methods

We compare against the following baseline methods in our experiments: i) L2A (Du et al., 2017), ii) Transformer (w/ copy) (Vaswani et al., 2017), and iii) MHQG+AE (Kumar et al., 2019). To the best of our knowledge, Kumar et al. (2019) is probably the first NN-based work that focused on the same setting as ours. Their proposed model, called MHQG+AE, employs a Transformer-based encoder to encode a KG subgraph (i.e., a set of triples), and generates an output question with a Transformer-based decoder. L2A is a LSTM-based Seq2Seq model equipped with attention mechanism that was originally designed for the QG from text task where the input is a sequence of tokens. Kumar et al. (2019) included L2A as a baseline by feeding it the linearised KG subgraph. The results of L2A reported here are taken from Kumar et al. (2019). We also include a Transformer-based encoder-decoder model that takes as input the linearised KG subgraph, i.e., a sequence of triples where each triple is represented as a sequence of tokens containing the subject name, predicate name and object name in order. We used the open-source implementation (Klein et al., 2017) of the Transformer-based encoder-decoder model that is equipped with the copying mechanism.

4.2 Data and Metrics

Following Kumar et al. (2019), we used WebQuestions (WQ) and PathQuestions (PQ) as our benchmarks where both of them use Freebase (Google, 2018) as the underlying KG. The WQ dataset combines examples from WebQuestionsSP (Yih et al., 2016) and ComplexWebQuestions (Talmor and Berant, 2018) where both of them are question answering datasets that contain natural language questions, corresponding SPARQL queries and answer entities. For each instance in WQ, in order to construct the KG subgraph, Kumar et al. (2019) converted its SPARQL query to return a subgraph instead of the answer entity, by changing it from a SELECT query to a CONSTRUCT query. The WQ dataset contains 18,989/2,000/2,000 (train/development/test) examples. The PQ dataset (Zhou et al., 2018) is similar to WQ except that the KG subgraph in PQ is a path between two entities that span two or three hops. The PQ dataset contains 9,793/1,000/1,000 (train/development/test) examples. Brief statistics of the two datasets are provided in Table 1.

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

4.3 Experimental Results

Table 2 shows the evaluation results comparing our proposed models against other state-of-the-art baseline methods on WQ and PQ test sets. As we can see, our models outperform all baseline methods by a large margin on both benchmarks. Besides, we can clearly see the advantages of GNN-based encoders for modeling KG subgraphs, by comparing our model with RNN-based (i.e., L2A) and Transformer-based (i.e., Transformer, MHQG+AE) baselines. Compared to our Graph2Seq model, both RNN-based and Transformer-based baselines ignore the explicit graph structure of a KG subgraph, which leads to degraded performance. Interestingly, the Transformer baseline performs reasonably well on PQ, but dramatically fails on WQ. We speculate this is because PsQ is more friendly to sequential models such as Transformer as the KG subgraph in PQ is more like path-structure while the one in WQ is more like tree-structure.

Data	# entities	# predicates	# triples	# instances	query length
WQ	25,703	672	2/99/5.8	22,989	5/36/15
PQ	7,250	378	2/3/2.7	9,731	8/25/14

Table 1: Data statistics. The min/max/avg numbers of triples contained in KG subgraphs and the min/max/avg query lengths are reported.

We also compare two variants (i.e., G2S vs. G2S$_{edge}$) of our model for handling multi-relational graphs. As shown in Table 2, directly applying the BiGGNN encoder to a Levi graph which is converted from a KG subgraph works quite well. The proposed BiGGNN$_{edge}$ model can directly handle multi-relational graphs without modifying the input graph. However, it performs slightly worse than the Levi graph solution. We suspect that it might help improve the modeling power of BiGGNN$_{edge}$ by updating edge embeddings in the message passing process and attending to edges in the attention mechanism. Currently, these two features are missing in BiGGNN$_{edge}$. We leave these as future work.

4.4 Model Analysis

4.4.1 Ablation Study

We perform an ablation study to examine the performance impacts of different model components, as shown in Table 3. First of all, the node-level copying mechanism contributes a lot to the overall model performance. By turning it off, we observe significant performance drops on both benchmarks. This verifies our assumption that when generating questions from a KG subgraph, one usually directly copies named entities from the input KG subgraph to the output question. Besides, the answer information is also important for generating relevant questions. Even with the simple answer markup technique, we can see the performance boost on both benchmarks.

4.4.2 Effect of Node/Edge Embedding Initialization

We empirically compare two different ways of initializing the embeddings of nodes and edges of the KG subgraph when applying the Graph2Seq model. As shown in Table 4, encoding nodes and edges based on word embeddings of their textual names works better than based on their KG embeddings. We suspect this is because it is difficult for a NN-based model to learn the gap between KG embeddings on the encoder side and word embeddings on the decoder side. With the word embedding-based encoding strategy, it is relatively easier for a model to learn the mapping from the input KG subgraph to the output natural language question. It also seems that modeling local dependency within the subgraph without utilizing the global KG information is enough for generating meaningful questions from a KG subgraph.

4.4.3 Effect of the Number of GNN Hops

Fig. 2 shows the impact of the number of GNN hops when applying a GNN-based encoder to encode the KG subgraph in WQ. It indicates that increasing the number of GNN hops can boost the model performance until reaches some optimal value.

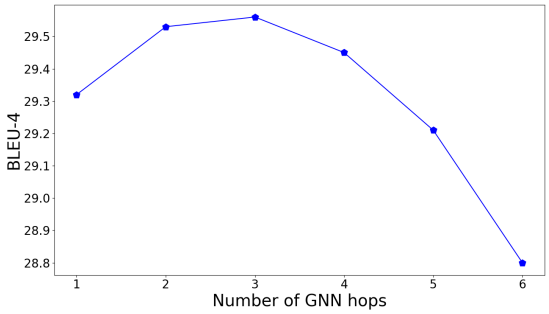

Figure 2: Effect of the number of GNN hops for G2S+AE on PQ.

4.4.4 Impact of Directionality on GNN Encoder

We also compare the performance of bidirectional Graph2Seq with unidirectional (i.e., forward and backward) Graph2Seq. As show in Table 5, performing only one direction message passing when processing the KG subgraph degrades the model performance.

4.5 Results on Training the Model with a Hybrid Objective

Table 6 and Table 7 show the results of training our proposed G2S+AE model with a hybrid objective combining both cross-entropy loss and RL loss. While the RL-based training strategy boosts the

Method	WQ			PQ		
	BLEU-4	METEOR	ROUGE-L	BLEU-4	METEOR	ROUGE-L
L2A	6.01	25.24	26.95	17.00	19.72	50.38
Transformer	8.94	13.79	32.63	56.43	43.45	73.64
MHQG+AE	11.57	29.69	35.53	25.99	33.16	58.94
G2S+AE	**29.45**	30.96	**55.45**	**61.48**	44.57	**77.72**
G2S$_{edge}$ +AE	29.40	**31.12**	55.23	59.59	**44.70**	75.20

Table 2: Evaluation results on WQ and PQ.

Method	WQ			PQ		
	BLEU-4	METEOR	ROUGE-L	BLEU-4	METEOR	ROUGE-L
G2S+AE	29.45	30.96	55.45	61.48	44.57	77.72
G2S	28.43	30.13	54.44	60.68	44.07	75.94
G2S w/o copy	22.95	26.99	51.05	57.10	42.66	74.29

Table 3: Ablation study on WQ and PQ.

Method	BLEU-4	METEOR	ROUGE-L
w/ word emb.	28.43	30.13	54.44
w/ KG emb.	22.80	25.85	48.93

Table 4: Effect of node/edge init embeddings for G2S on WQ.

Method	BLEU-4	METEOR	ROUGE-L
Bidirectional	61.48	44.57	77.72
Forward	59.59	42.72	75.82
Backward	59.12	42.66	75.03

Table 5: Impact of directionality for G2S+AE on PQ test set.

model performance on WQ, it does not help the model training on PQ.

Method	BLEU-4	METEOR	ROUGE-L
G2S+AE	29.45	30.96	55.45
G2S+AE+RL	29.80	31.29	55.51

Table 6: Results of RL-based G2S+AE on WQ.

Method	BLEU-4	METEOR	ROUGE-L
G2S+AE	61.48	44.57	77.72
G2S+AE+RL	59.21	44.47	77.35

Table 7: Results of RL-based G2S+AE on PQ.

4.6 Case Study

As shown in Table 8, we conduct case study to examine the quality of generated questions using different ablated systems. First of all, by initializing node/edge embeddings with KG embeddings, the model fails to generate reasonable questions. Besides, with the node-level copying mechanism, the model is able to directly copy the entity name "giza necropolis" into the output question. Last, incorporating the answer information helps genrate more relevant and specific questions.

KG subgraph: (Egypt, administrative_divisions, Cairo), (Giza Necropolis, containedby, Egypt)
Gold: what country has the city of cairo and is home of giza necropolis ?
G2S w/ KG emb.: what country that contains cairo has cairo as its province ?
G2S w/o copy: where is the giza giza located in that has cairo ?
G2S: where is the giza necropolis located in that contains cairo ?
G2S+AE: what country that contains cairo is the location of giza necropolis ?

Table 8: Generated questions on WQ test set. Target answers are underlined. For the sake of brevity, we only display the lowest level of the predicate hierarchy.

5 Conclusion

We introduced a novel bidirectional Graph2Seq model to generate natural language questions from a KG subgraph and target answers. Novel node-level copying mechanism was proposed to allow di-

rectly copying node attributes from the input graph to the output question. We also explored different ways of initializing node/edge embeddings and handling multi-relational graphs. Our model achieves new state-of-the-art scores, outperforming existing methods by a significant margin on the two benchmarks.

Future directions include combining both word and KG embeddings for node/edge embedding initialization, and exploring effective ways of utilizing answer information.

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.

Junwei Bao, Duyu Tang, Nan Duan, Zhao Yan, Yuanhua Lv, Ming Zhou, and Tiejun Zhao. 2018. Table-to-text: Describing table region with natural language. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. 2017. Graph convolutional encoders for syntax-aware neural machine translation. *arXiv preprint arXiv:1704.04675*.

Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-sequence learning using gated graph neural networks. *arXiv preprint arXiv:1806.09835*.

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.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Yu Chen, Lingfei Wu, and Mohammed J Zaki. 2019a. Bidirectional attentive memory networks for question answering over knowledge bases. *NAACL*.

Yu Chen, Lingfei Wu, and Mohammed J Zaki. 2019b. Deep iterative and adaptive learning for graph neural networks. *arXiv preprint arXiv:1912.07832*.

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

Yu Chen, Lingfei Wu, and Mohammed J Zaki. 2020. Reinforcement learning based graph-to-sequence model for natural question generation. *ICLR*.

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.

Bayu Distiawan, Jianzhong Qi, Rui Zhang, and Wei Wang. 2018. Gtr-lstm: A triple encoder for sentence generation from rdf data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1627–1637.

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 Laforest. 2018. Zero-shot question generation from knowledge graphs for unseen predicates and entity types. *arXiv preprint arXiv:1802.06842*.

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.

Google. 2018. Freebase data dumps. https://developers.google.com/freebase.

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.

Steven Haussmann, Oshani Seneviratne, Yu Chen, Yarden Neeman, James Codella, Ching-Hua Chen, Deborah L McGuinness, and Mohammed J Zaki. 2019. Foodkg: A semantics-driven knowledge graph for food recommendation. In *International Semantic Web Conference*, pages 146–162. Springer.

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

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*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810*.

Vishwajeet Kumar, Yuncheng Hua, Ganesh Ramakrishnan, Guilin Qi, Lianli Gao, and Yuan-Fang Li. 2019. Difficulty-controllable multi-hop question generation from knowledge graphs. In *International Semantic Web Conference*, pages 382–398. Springer.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. *arXiv preprint arXiv:1603.07771*.

Friedrich Wilhelm Levi. 1942. *Finite geometrical systems: six public lectues delivered in February, 1940, at the University of Calcutta*. The University of Calcutta.

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*.

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

Diego Marcheggiani and Laura Perez-Beltrachini. 2018. Deep graph convolutional encoders for structured data to text generation. *arXiv preprint arXiv:1810.09995*.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

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.

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.

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

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 European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 376–385.

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*.

Dominic Seyler, Mohamed Yahya, and Klaus Berberich. 2015. Generating quiz questions from knowledge graphs. In *Proceedings of the 24th International Conference on World Wide Web*, pages 113–114. ACM.

Dominic Seyler, Mohamed Yahya, and Klaus Berberich. 2017. Knowledge questions from knowledge graphs. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*, pages 11–18. ACM.

Martin Simonovsky and Nikos Komodakis. 2017. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3693–3702.

Linfeng 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.

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

Linfeng Song and Lin Zhao. 2016. Question generation from a knowledge base with web exploration. *arXiv preprint arXiv:1610.03807*.

I Sutskever, O Vinyals, and QV Le. 2014. Sequence to sequence learning with neural networks. *Advances in NIPS*.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. *arXiv preprint arXiv:1803.06643*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

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

Pavlos Vougiouklis, Hady Elsahar, Lucie-Aimée Kaffee, Christophe Gravier, Frederique Laforest, Jonathon Hare, and Elena Simperl. 2018. Neural wikipedian: Generating textual summaries from knowledge base triples. *Journal of Web Semantics*, 52:1–15.

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

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*.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 974–983. ACM.

Mantong Zhou, Minlie Huang, and Xiaoyan Zhu. 2018. An interpretable reasoning network for multi-relation question answering. *arXiv preprint arXiv:1801.04726*.

A Hybrid Evaluator

Most prior works on QG applied the cross-entropy based training objective, which is also a de facto choice for training sequential models in many other NLP tasks. However, cross-entropy based training strategy has some known limitations including exposure bias and evaluation discrepancy between training and testing (Ranzato et al., 2015; Wu et al., 2016; Paulus et al., 2017). To tackle these issues, we introduce a hybrid objective function combining both cross-entropy loss and Reinforcement Learning (RL) (Williams, 1992) loss for training our Graph2Seq model.

Specifically, we introduce a two-stage training strategy as follows. In the first stage, the regular cross-entropy loss is used,

$$\mathcal{L}_{lm} = \sum_{t} -\log P(y_t^*|X, y_{<t}^*) \tag{7}$$

where y_t^* is the word at the t-th position of the ground-truth output sequence. Scheduled teacher forcing (Bengio et al., 2015) is adopted to alleviate the exposure bias problem. In the second stage, we further fine-tune the model by optimizing a mixed objective function combining both cross-entropy loss and RL loss, defined as,

$$\mathcal{L} = \gamma\mathcal{L}_{rl} + (1-\gamma)\mathcal{L}_{lm} \tag{8}$$

where γ is a scaling factor controling the trade-off between the two losses. During the testing phase, beam search is applied to generate the output.

While our architecture is agnostic to the specific REINFORCE algorithm, in this work, we employ an efficient yet effective REINFORCE algorithm, called self-critical sequence training (SCST) (Rennie et al., 2017) , to directly optimize the discrete evaluation metrics. At each training iteration, the RL loss is defined by comparing the reward of the sampled output Y^s with the reward of the baseline output $\hat{Y}$,

$$\mathcal{L}_{rl} = (r(\hat{Y}) - r(Y^s)) \sum_{t} \log P(y_t^s|X, y_{<t}^s) \tag{9}$$

where Y^s is produced by multinomial sampling, that is, each word y_t^s is sampled according to the likelihood $P(y_t|X, y_{<t})$ predicted by the generator, and $\hat{Y}$ is obtained by greedy search, that is, by maximizing the output probability distribution at each decoding step. As we can see, minimizing the above loss is equivalent to maximizing the likelihood of some sampled output that has a higher reward than the corresponding baseline output.

One of the key factors for RL is to pick the proper reward function. We define $r(Y)$ as the reward of an output sequence Y, computed by comparing it to the corresponding ground-truth sequence Y^* with some reward metric which is a combination of our evaluation metrics (e.g., BLEU-4, ROUGE-L, etc.). This lets us directly optimize the model towards the evaluation metrics.

B Model Settings

We keep and fix the 300-dim GloVe (Pennington et al., 2014) vectors for those words that occur more than twice in the training set. The dimensions of answer markup embeddings are set to 32 and 24 for WQ and PQ, respectively. We set the hidden state size of BiLSTM to 150 so that the concatenated state size for both directions is 300. The size of all other 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. The label smoothing ratio is set to 0.2. The number of GNN hops is set to 4. During training, in each epoch, we set the initial teacher forcing probability to 0.8 and exponentially increase it to $0.8 * 0.9999^i$ where i is the training step. We use Adam (Kingma and Ba, 2014) as the optimizer. The learning rate is set to 0.001 in the pretraining stage. In the fine-tuning stage, we set the learning rate to 0.00001 and 0.00002 for WQ and PQ, respectively. 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 30. The beam search width is set to 5. All hyperparameters are tuned on the development set.