Toward Subgraph-Guided Knowledge Graph Question Generation With Graph Neural Networks

Yu Chen, Lingfei Wu, *Member, IEEE*, and Mohammed J. Zaki, *Fellow, IEEE*

***Abstract*— Knowledge graph (KG) question generation (QG) aims to generate natural language questions from KGs and target answers. Previous works mostly focus on a simple setting that is 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 previous works built on either RNN- or Transformer-based models to encode a linearized KG subgraph, which totally discards the explicit structure information of a KG subgraph. To address this issue, we propose to apply a bidirectional Graph2Seq model to encode the KG subgraph. Furthermore, we enhance our RNN decoder with a node-level copying mechanism to allow direct copying of node attributes from the KG subgraph to the output question. Both automatic and human evaluation results demonstrate that our model achieves new state-of-the-art scores, outperforming existing methods by a significant margin on two QG benchmarks. Experimental results also show that our QG model can consistently benefit the question-answering (QA) task as a means of data augmentation.**

***Index Terms*— Deep learning, graph neural networks (GNNs), knowledge graphs (KGs), natural language (NL) processing, question generation (QG).**

I. Introduction

RECENT years have seen a surge of interest in question generation (QG) in machine learning and natural language (NL) processing. The goal of QG is to generate an NL question for a given form of data such as text [1], [2], [3], [4], images [5], tables [6], and knowledge graphs (KGs) [7]. In this work, we focus on QG from KGs.

One of the biggest applications of QG is to provide training data for question-answering (QA) systems [8]. KGs have drawn a large amount of research attention in recent years, partially due to their huge potential for an accessible, natural way of retrieving information without a need for learning complex query languages such as SPARQL. In order to train a large knowledge base QA (KBQA) system [9], [10], a large number of labeled question–answer pairs are often needed, which can be a severe bottleneck in practice because human annotation is usually expensive and time-consuming. Developing effective approaches to generate high-quality QA pairs can significantly address the data scarcity issue for KBQA. In addition, QG can be applied for educational purposes by producing practice assessments [11]. Moreover, QG can help dialogue systems have more engaging conversations [12].

In the past decade, the research on QG from KGs has gained increasing interest and can be categorized into two classes. The first line of research heavily relies on handcrafted question templates [7], [13], [14]. They typically first construct a structured query (e.g., SPARQL query) and then apply a template-based method to verbalize it to an NL question. Using a set of predesigned templates not only requires a significant amount of human effort, thus leading to low generalizability and scalability, but also limits the diversity and fluency of the generated questions. The other line of research adopts a purely data-driven end-to-end approach without resorting to any handcrafted templates. Those are mostly neural network-based approaches that employ an RNN or Transformer [15] decoder to generate an NL question as a sequence of tokens. However, most of them [16], [17], [18], [19], [20] focus only on generating simple questions, which limits their usage in benefiting complex KBQA systems often requiring multihop reasoning. The main reason why they can only generate simple questions is due to their incapability of encoding a KG subgraph containing a rich set of interlinked triples. Instead, they can only take a keyword list or a single KG triple (i.e., subject–predicate–object) as the input because they adopt a sequence-to-sequence (Seq2Seq) [21], [22] architecture which can only encode sequential data via a sequence encoder.

More recently, [23] presented a Transformer-based Seq2Seq model named MHQG + AE for generating multihop complex questions from a KG subgraph. To the best of our knowledge, MHQG + AE was the first neural network-based model focusing on QG from a KG subgraph. Because a Transformer cannot admit graph-structured input data like a KG subgraph, they proposed to represent a KG subgraph as a set of triples where the triple embeddings were computed based on the embeddings of the subject, predicate, and object contained in the triple. They also removed the positional encoding in a regular Transformer in order to discard the position information of triples in a KG subgraph. Even though their approach was able to directly work on a KG subgraph for generating more complex questions compared to previous approaches, they failed to effectively utilize the rich structure information of a KG subgraph because they completely ignored the rich interactions among triples in a KG subgraph.

Manuscript received 5 October 2020; revised 28 August 2021, 12 March 2022, and 31 July 2022; accepted 21 March 2023. This work was supported by IBM Research AI through the AI Horizons Network under the RPI-IBM HEALS Project. *(Corresponding author: Lingfei Wu.)*

Yu Chen is with Meta AI, Menlo Park, CA 94025 USA (e-mail: hugochen@meta.com).

Lingfei Wu is with Pinterest, San Francisco, CA 94107 USA (e-mail: teddy.lfwu@gmail.com).

Mohammed J. Zaki is with the Computer Science Department, Rensselaer Polytechnic Institute, Troy, NY 12180 USA (e-mail: zaki@cs.rpi.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2023.3264519.

Digital Object Identifier 10.1109/TNNLS.2023.3264519

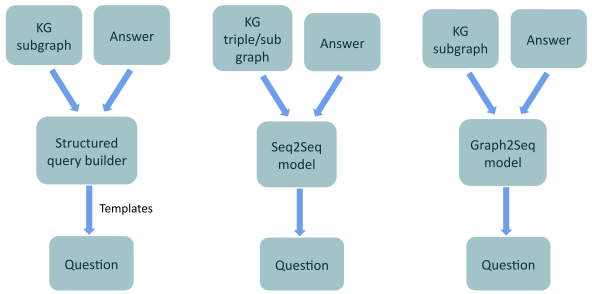


Fig. 1. Various QG from KG learning paradigms. Left: template-based approaches. Middle: Seq2Seq-based approaches for simple QG from a single KG triple or multihop QG from a KG subgraph. Right (ours): Graph2Seq-based approaches for multihop QG from a KG subgraph.

In follow-up work, Bi et al. [24] proposed to augment the input KG subgraph with external knowledge such as entity descriptions/domains, question word types, and answer entity types. However, they still failed to respect the rich structure information of the KG subgraph as they simply regarded a KG subgraph as a sequence of subject, predicate, and object embeddings and applied a bidirectional LSTM [25] to learn its representations. We believe capturing fine-grained structure information is critical for generating high-quality questions.

We summarize the three challenges of the task of multihop QG from KGs (denoted as KG–QG) as follows. The first one is how to learn a good representation of a KG subgraph. A KG subgraph has complex underlying structures such as node attributes and multirelational edges. Each node and edge could have (long) associated text comprising multiple words. Previous approaches either only considered a keyword list or single triple for simple QG or simply regarded the KG subgraph as a set of triples without fully utilizing its rich structure information when generating multihop questions. The second challenge is how to automatically learn a good mapping between a subgraph and an NL question. For instance, it is common for a question to directly mention an entity name from the input KG subgraph. However, it is challenging for previous approaches to precisely generate such an entity name which often contains multiple tokens. The third challenge is how to effectively leverage the answer information. Given a KG subgraph containing many triples, one can generate completely different questions without knowing the exact target answer. Therefore, effectively utilizing the answer information is crucial for generating more relevant questions.

In order to address the above challenges, we present a subgraph-guided KG–QG approach with graph neural networks (GNNs). To this end, we introduce, for the first time, the Graph-to-Sequence (Graph2Seq) architecture with a novel node-level copying mechanism for the KG–QG task to address the second challenge. We extend the regular GNN-based encoder to allow processing directed and multirelational KG subgraphs to solve the first challenge. In addition, we propose a simple, yet elegant way to leverage the context information from the answers to effectively handle the third challenge. Extensive experimental results demonstrate that our model significantly outperforms the state-of-the-art baselines by a large margin on two benchmarks and consistently benefits the KBQA task. Fig. 1 illustrates the main ideas of various QG from KG learning paradigms.

We highlight our main contributions as follows.

1) We propose a novel Graph2Seq model for subgraph-guided KG–QG. The proposed Graph2Seq model employs bidirectional graph embedding and we design two different GNN encoders to effectively encode KG subgraphs with directed and multirelational edges.
2) We extend the RNN decoder with a novel copying mechanism that allows the entire node attribute to be borrowed from the input KG subgraph when generating NL questions.
3) We investigate two different ways of initializing node/edge embeddings when applying a GNN encoder to process KG subgraphs. In addition, we study the impact of edge direction on the GNN encoder.
4) Experimental results show that our model improves the state-of-the-art BLEU-4 score from 11.57 to 29.40 and from 25.99 to 59.59 on WebQuestions (WQ) and PathQuestions (PQ) benchmarks, respectively. A human evaluation study corroborates that the questions generated by our model are more natural (semantically and syntactically) and relevant compared to other baselines. Experiments also show that our QG model can consistently benefit the KBQA task as a means of data augmentation.

II. Related Work

A. QG From KGs

Early works [7], [13], [14] on QG from KGs are mostly template-based approaches heavily relying on a set of predefined question templates to verbalize a structured query to an NL question. However, they usually have low generalizability and scalability, and the diversity and fluency of the generated questions are limited due to the nature of template-based approaches. Recently, Seq2Seq-based neural architectures have been applied to this task without resorting to manually designed templates and are end-to-end trainable. However, these approaches [16], [17], [18], [19], [20] only focus on generating simple questions from a keyword list or single triple as they typically employ an RNN- or Transformer-based encoder that cannot handle graph-structured data like a KG subgraph. Very recently, Seq2Seq-based approaches were also applied for generating a multihop complex question from a KG subgraph instead of just a single triple. However, they still failed to effectively utilize the rich structure information of the KG subgraph by simply regarding a KG subgraph as a set of triples [23] or a sequence of subject, predicate, and object embeddings [24]. Unlike all previous approaches, in this work, we focus on generating multihop complex questions by effectively modeling the rich structures (e.g., edge directions and edge types) of KG subgraphs via a novel GNN-based graph encoder. To the best of our knowledge, we are the first to introduce the Graph2Seq architecture to the KG–QG task.

There was related work focusing on QG from the text. In [3], we proposed a reinforcement learning (RL)-based Graph2Seq model for the task of QG from the text. Besides the difference in terms of problem settings, the major technical difference between this work and our previous work includes, in this work: 1) we extend the GNN encoder to handle multirelational graphs where in [3] edge-type information was not

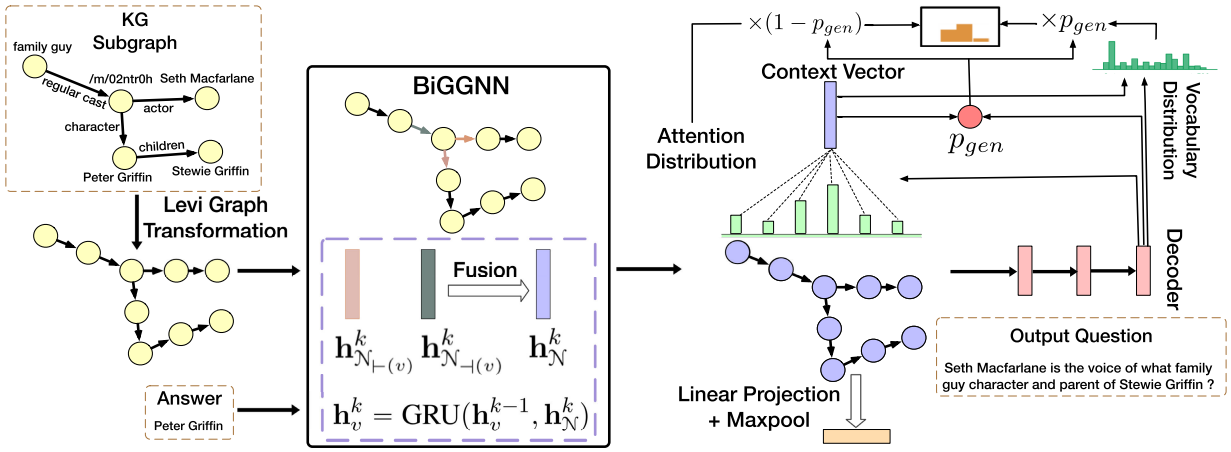


Fig. 2. Overall architecture of our proposed model. Best viewed in color.

modeled and 2) we extend the word-level copying mechanism in [3] to the node-level copying mechanism. Some recent QG from text works explored leveraging external knowledge for better performance. For instance, Shen et al. [26] proposed to augment the raw text with auxiliary knowledge retrieved from a KG using entities and keywords mentioned in the input text. Their approach then applies three different GNN-based encoders to encode three types of graphs constructed based on text and knowledge retrieved from a KG. Even though we both adopt a Graph2Seq architecture, we tackle very different problems and they utilize KG as external knowledge for better QG from text performance.

Our work is also related to recent research efforts on pretrained models for KG-to-text generation [27], [28] which used KG-to-text generation as one of the pretraining tasks. These large-scale pretrained models could be used for many downstream KG-to-text applications (including QG from KGs) by finetuning them for a particular downstream task.

B. Graph Neural Networks

Traditional deep-learning approaches like convolutional neural networks and recurrent neural networks are designed for Euclidean data like images and text and thus cannot directly handle non-Euclidean data like graphs. Over the past few years, GNNs [29], [30], [31], [32], [33], [34] have drawn increasing attention due to their ability to model graph-structured data and have successfully been applied in the NLP field [35], [36], [37], [38], [39]. Recently, in order to address the limitations of the widely used Seq2Seq architectures [21], [22] on encoding rich and complex graph-structured data, a number of works have applied the Graph2Seq architectures for various NLP tasks including machine translation [35], [40], semantic parsing [41], code summarization [42], and graph-to-text generation (e.g., AMR, SQL, and KG to text) [43], [44], [45], [46]. Compared to existing Graph2Seq models, our proposed Graph2Seq model can better handle multirelational graphs and employ a node-level copying mechanism to enable generating more faithful text.

III. Approach

A. Problem Formulation

Our focus is on natural QG from a KG subgraph, along with potential target answers; the overall architecture of our approach is shown in Fig. 2. 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 $V \subseteq \mathcal{V}$ denotes a set of entities (i.e., subjects or objects) and $E \subseteq \mathcal{E}$ 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 [9]. The task of KG–QG is to generate the best NL question consisting of a sequence of word tokens $\hat{Y} = \{y_1, y_2, \ldots, y_T\}$ that maximizes the conditional likelihood $\hat{Y} = \arg\max_Y P(Y|\mathcal{G}, V^a)$, where T is the length of the question. We focus on the problem setting where we have a set of KG subgraphs (and answers) and target questions pairs, to learn the mapping; existing QG approaches [16], [18], [23] make a similar assumption. Although the three main challenges we have discussed before are based on QG from KGs, other QG tasks from other data sources also share some or most of the issues when dealing with these tasks. Therefore, our model could be generalized to cope with these tasks as well.

B. Encoding Layer

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

1) Encoding Nodes and Edges: Before applying the GNN encoder to process a KG subgraph, we need to map nodes and edges to an 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 KG representation learning algorithm such as TransE [47], while the other one is based on pretrained embeddings (e.g., GloVe [48]) of the 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 posit that it is relatively easier for a model to learn the mapping from the input KG subgraph to the output NL question with both sides based on word embeddings. We empirically compare and analyze the two encoding strategies in our

experiments. In order to encode the nodes and edges in a KG subgraph, we apply two bidirectional LSTMs [25] for nodes (i.e., one for nodes and one for edges) to encode their associated text. The concatenation of the last forward and backward hidden states of the BiLSTM is used as the initial embeddings for nodes and edges.

2) Utilizing Target Answers: In the setting of KBQA [9], [49], it is usually assumed that the answers to a question are entities in a KG subgraph. As a dual task of KBQA, in this QG work, we assume that utilizing the target answers along with the KG subgraph can help generate more relevant questions. To this end, we apply a simple, yet effective strategy where we introduce an additional learnable markup vector associated with each node/edge to indicate whether it is an answer or not. Therefore, the initial vector representation of a node/edge will be the concatenation of the BiLSTM output and the answer markup vector. We denote $\mathbf{X}^e = \{\mathbf{x}_1^e, \mathbf{x}_2^e, \ldots, \mathbf{x}_n^e\}$ and $\mathbf{X}^p = \{\mathbf{x}_1^p, \mathbf{x}_2^p, \ldots, \mathbf{x}_m^p\}$ as the embeddings of the entity nodes and predicate edges, respectively. Both $\mathbf{X}^e$ and $\mathbf{X}^p$ have the same embedding dimension d.

C. Bidirectional Graph2Seq Generator With Copying

While RNNs are good at modeling sequential data, they cannot naturally handle graph-structured data. One might need to linearize a graph to a sequence to apply an RNN-based encoder, which will lose the rich structure information in the graph. Many previous works [35], [40] showed the superiority of GNNs compared to RNNs in modeling graph-structured data. Kumar et al. [23] proposed to encode a set of triples via a Transformer by removing positional encoding in the original architecture. Even though a Transformer-based encoder could learn the semantic relations among the triples through all-to-all attention, the explicit graph structure is totally discarded. In this work, we introduce a bidirectional GNN-based encoder to encode the KG subgraph and decode the output question via an RNN-based decoder equipped with a node-level copying mechanism.

1) Bidirectional Graph Encoder: Many existing GNNs [29], [31], [50] were not designed to process directed graphs such as a KG. Even though some GNN variants such as GGSNN [32] and MPNN [30] 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 GNN (BiGGNN) which extends GGSNN by learning node embeddings from both incoming and outgoing directions in an interleaved fashion when processing a directed graph. A similar bidirectional approach has been exploited in [43] and [51] to extend other GNN variants. While their methods simply learn the node embeddings of each direction independently and concatenate them at the last 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 $\mathbf{x}_v$, namely, a concatenation of the BiLSTM output and the answer markup vector. BiGGNN then performs message passing across the graph 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 [50] can be employed to aggregate neighborhood information. Here, we use a simple average aggregator

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

where $\mathcal{N}_{\dashv(v)}$ and $\mathcal{N}_{\vdash(v)}$ denote the incoming and outgoing neighbors of node v. We then fuse the node embeddings aggregated from both directions

$$\mathbf{h}_{\mathcal{N}_{(v)}}^k = \mathrm{Fuse}\left(\mathbf{h}_{\mathcal{N}_{\dashv(v)}}^k, \mathbf{h}_{\mathcal{N}_{\vdash(v)}}^k\right). \tag{2}$$

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

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

where $\odot$ is the componentwise multiplication, σ is a sigmoid function, and $\mathbf{z}$ is a gating vector. The gate helps the model determine how much of the information needs to be reserved from the two aggregated node embeddings.

Finally, a gated recurrent unit (GRU) [22] is used to update the node embeddings by incorporating the aggregation information

$$\mathbf{h}_v^k = \mathrm{GRU}\left(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}_{(v)}}^k\right). \tag{4}$$

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 $\mathbf{h}^{\mathcal{G}}$.

2) Handling Multirelational Graphs: KGs are typically heterogeneous networks that contain a large number of edge types. However, many existing GNNs [29], [31], [32], [50] are not directly applicable to multirelational graphs. In order to model both node and edge information with GNNs, researchers have extended them by either having separate learnable weights for different edge types or having explicit edge embeddings when performing message passing [30], [52]. While the former solution may have severe scalability issues when handling graphs with a large number of edge types, the latter requires major modifications to existing GNNs. In this work, we explore two solutions to adapt GNNs to multirelational graphs.

a) Levi graph transformation: We can directly apply regular GNNs to a multirelational KG subgraph by converting it to a Levi graph [53]. Specifically, we treat all edges in the original graph as new nodes and add new edges connecting the 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) will be converted to "Mario_Siciliano $\rightarrow$ place_of_birth $\rightarrow$ Rome," where "place_of_birth" becomes a new node, and $\rightarrow$ 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 at most be linear to the number of original nodes.

b) Gated message passing with edge information: We also extend BiGGNN to explicitly incorporate edge embeddings when conducting message passing, calling the resultant variant as $\text{BiGGNN}_{\text{edge}}$. Specifically, we rewrite the node aggregation function (1) as follows:

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

where f is a nonlinear function (i.e., linear projection + ReLU [54]) applied to the concatenation of $\mathbf{h}^{k-1}_u$ and $\mathbf{e}_{uv}$ which is the embedding of the edge connecting node u and v.

3) RNN Decoder With Node-Level Copying: We adopt an attention-based [55], [56] 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 [57]. 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 $\mathbf{s}_t$ and the attention memory.

We hypothesize that when generating NL 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 [58], [59], most RNN decoders are typically allowed to copy words from the input sequence. We extend the regular word-level copying mechanism to the node-level copying mechanism that allows copying node attributes (i.e., node text) from the input graph. The copying mechanism was used in some previous Graph2Seq articles [45], [60]. The most similar work is [60] which proposed to copy both entities and predicates from the input graph. Unlike [60], we use a masked copying mechanism to only copy entity nodes in the transformed Levy graph and do not copy predicate nodes. This is because we assume that for the KG-QG task, it is very likely for humans to directly mention entity names but not necessarily for predicate names that are from the KG.

At each decoding step, 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. We dynamically maintain an extended vocabulary which is the union of the usual vocabulary and all node names appearing in a batch of source examples (i.e., KG subgraphs).

D. Training and Testing

As customary for training sequential models, we minimize the following cross-entropy loss:

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

where y^*_t is the word at the tth position of the gold output sequence. Scheduled teacher forcing [61] is adopted to alleviate the exposure bias problem. During the testing phase, beam search is applied to generate the output.

Two-Stage Training Strategy: Most prior works on QG employ cross-entropy-based training objectives, 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 [62], [63], [64]. That is to say, during training, a model has access to the ground-truth previous token when decoding and is optimized toward cross-entropy loss, while during testing, no ground-truth previous token is provided and cross-entropy loss is not used for evaluation.

To tackle these issues, besides training our proposed model with the regular cross-entropy loss, we also explore a two-stage training strategy where we first train the model with cross-entropy loss and then finetune the model with a hybrid loss combining both the cross-entropy loss and RL [65] loss. The RL loss is defined based on evaluation metrics, enabling us to directly optimize the model toward the evaluation metrics.

The reason we need the first-stage training is that training models from scratch using RL is often challenging. The regular cross-entropy training can help us obtain a reasonably good performing model, and the RL-based finetuning can further improve the model performance.

In the first stage, regular cross-entropy loss is used

$$\mathcal{L}_{\text{lm}} = \sum_t -\log P\big(y^*_t | X, y^*_{<t}\big) \quad (7)$$

as in (6). In the second stage, we further finetune the model by optimizing a hybrid objective function combining both cross-entropy loss and RL loss, defined as

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

where γ is a scaling factor controlling the tradeoff between the two losses.

While our architecture is agnostic to the specific RL algorithm, in this work, we employ an efficient, yet effective RL approach called self-critical sequence training (SCST) [66] to directly optimize the discrete evaluation metrics. SCST is an efficient REINFORCE algorithm that utilizes the output of its own test-time inference algorithm to normalize the rewards it experiences. 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 $\hat{Y}$

$$\mathcal{L}_{\text{rl}} = \big(r\big(\hat{Y}\big) - r\big(Y^s\big)\big) \sum_t \log P\big(y^s_t | X, y^s_{<t}\big) \quad (9)$$

where Y^s is produced by multinomial sampling, that is, each word y^s_t 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.

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 (i.e., we used BLEU-4 and ROUGE-L scores in our experiments). This lets us directly optimize the model toward the evaluation metrics.

IV. Experiments

In this section, we conduct extensive experiments to evaluate the effectiveness of our proposed model for the QG task. We also conduct experiments to examine whether our QG model can help the QA task by providing more training data. 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 multirelational 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 a bidirectional GNN-based encoder and a unidirectional GNN-based encoder is also provided. The code and data will be released upon this article's acceptance.

A. Baseline Methods

We compare our model against the following baselines: 1) L2A [1]; 2) transformer (w/copy) [15]; 3) MHQG + AE [23]; 4) JointGT (T5) [28]; and 5) JointGT (BART) [28]. To the best of our knowledge, MHQG + AE was probably the first neural network-based model that focused on QG from a KG subgraph. Their proposed model, called MHQG + AE, employs a Transformer-based encoder [15] to encode a KG subgraph (i.e., a set of triples) and generates an output question with a Transformer-based decoder. L2A is an LSTM-based Seq2Seq model equipped with an attention mechanism, which takes as input a linearized KG subgraph. It was included in [23] as a baseline. The results of L2A reported here are taken from [23]. We also include a Transformer-based encoder–decoder model [67] with a copying mechanism that takes as input a linearized KG subgraph, that is, a sequence of triples where each triple is represented as a sequence of tokens containing the subject name, predicate name, and object name. Hence, after the transformation, a KG subgraph becomes a sequence of tokens. Note that the Transformer baseline included in our experiments encodes the word sequence that is linearized from a KG subgraph, while the MHQG + AE model encodes the triple set contained in a KG subgraph by removing the positional encoding in a regular Transformer architecture. Unlike MHQG + AE that takes as input a set of triple embeddings that are pretrained by a knowledge-based representation learning framework called TransE [47], the Transformer baseline takes a sequence of word embeddings as input. We used the open-source implementation [67] of the Transformer-based encoder–decoder model that is equipped with a copying mechanism. Lastly, we also include two large-scale pretrained KG-to-text models JointGT (T5) and JointGT (BART) [28] which were finetuned for the task of QG from KGs. We do not include [24] as our baseline because their approach augmented the input KG subgraph with various types of external knowledge such as entity descriptions, entity domains, question word types, and answer entity types, which makes it unfair to directly compare the performance of their approach with our approach. In their original article, the authors reported that their ablated system without using auxiliary knowledge (i.e., it still utilized the additional question word type information, see the results in [24, Table 3]) significantly underperformed our approach (denoted as BiGraph2Seq in [24, Table 2]) on two benchmarks (i.e., 3.11 absolute BLEU-4 gap and 0.64 absolute BLEU-4 gap).

TABLE I
Data Statistics. The Min/Max/Avg Statistics Are Reported on the Queries and KG Subgraph Triples

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

B. Data and Metrics

Following [23], we used WQ and PQ[1] as our benchmarks where both of them use Freebase [68] as the underlying KG. The WQ dataset combines examples from WebQuestionsSP [69] and ComplexWebQuestions [70] where both of them are KBQA benchmarks that contain NL questions, corresponding SPARQL queries and answer entities. For each instance in WQ, in order to construct the KG subgraph, [23] 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 [23] contains 18 989/2 000/2 000 (train/development/test) examples. The PQ dataset [71] 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 I.

Following previous works, we use BLEU-4 [72], METEOR [73], and ROUGE-L [74] as automatic evaluation metrics. Initially, BLEU-4 and METEOR were designed for evaluating machine translation systems and ROUGE-L was designed for evaluating text summarization systems. We also conduct a human evaluation study on WQ. Generated questions are rated (i.e., range 1–5) based on whether they are syntactically correct, semantically correct, and relevant to the KG subgraph. More specially, we conducted a small-scale (i.e., 50 random examples per system) human evaluation study on the WQ test set. We asked six human evaluators to give feedback on the quality of questions generated by a set of anonymized competing systems. In each example, given a KG subgraph, target answers, and anonymized system output, they were asked to rate the quality of the output by answering the following three questions: 1) is this generated question syntactically correct? 2) is this generated question semantically correct? and 3) is this generated question relevant to the KG subgraph and target answers? For each evaluation question, the rating scale is from 1 to 5 where a higher score means better quality (i.e., 1: poor, 2: marginal,

[1]https://github.com/liyuanfang/mhqg

3: acceptable, 4: good, 5: excellent). Responses from all evaluators were collected and averaged.

C. Model Settings

We keep and fix the 300-dim GloVe [48] 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 [75] 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. In addition, partial teacher forcing is adopted, which means that when generating a sequence, some steps can be teacher forced and some not. We use Adam [76] as the optimizer. The learning rate is set to 0.001. 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 ten epochs. We clip the gradient at length 10. The batch size is set to 30. The beam search width is set to 5. In the RL fine-tuning experiments, we set γ in the mixed loss function (8) to 0.02 for WQ and 0.07 for PQ. And the ratios of the BLEU-4 score and ROUGE-L score for computing the reward are set to 1 and 0.02, respectively. We set the learning rate to 0.00001 and 0.00002 for WQ and PQ, respectively. All hyperparameters are tuned on the development set. Experiments were conducted on a machine that has an Intel i7-2700K CPU and an Nvidia Titan Xp GPU with 16 GB RAM.

D. Experimental Results

1) Automatic Evaluation Results: Table II 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 QG baselines by a large margin on both benchmarks. This verifies the effectiveness of the proposed model. 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- and Transformer-based baselines ignore the explicit graph structure of a KG subgraph, which leads to degraded performance. Although RNNs are suitable for processing sequential data such as text, they are incapable of modeling graph-structured data such as a KG subgraph. To apply the RNN-based L2A model to a KG subgraph, Kumar et al. [23] linearized the graph to a sequence during preprocessing. However, this inevitably ignores the rich structure information in the graph. Recently, the Transformer [15] has become a good alternative to the RNN when processing sequential data. Even though a transformer might be able to learn the semantic relations among the sequence elements through all-to-all attention, the explicit graph structure of a KG subgraph is totally discarded by the model. Given these limitations, as shown in our experiments, both of the two Transformer-based Seq2Seq baselines significantly underperform our GNN-based Graph2Seq model. Interestingly, the Transformer baseline performs reasonably well on PQ, but dramatically fails on WQ. We speculate this is because PQ 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.

The comparisons with large-scale pretrained KG-to-text models further demonstrated the superiority of our models. Without access to a large amount of pretraining data, our best-performing model clearly outperforms the large-scale model JointGT (T5) and achieves competitive results compared to JointGT (BART). We also compare two variants of our model (i.e., G2S vs. G2S_{edge}) for handling multirelational graphs. As shown in Table II, directly applying the BiGGNN encoder to a Levi graph converted from a KG subgraph works quite well. The proposed $\text{BiGGNN}_{\text{edge}}$ model can directly handle multirelational graphs without modifying the input graph. However, it performs slightly worse than the Levi graph solution. Future directions of improving $\text{BiGGNN}_{\text{edge}}$ include updating edge embeddings in the message-passing process and attending to edges in the attention mechanism.

2) Human Evaluation Results: We conduct a human evaluation study to assess the quality of the questions generated by our model, the Transformer baseline, and the ground-truth data in terms of syntax, semantics, and relevance metrics. In addition, an overall score is computed for each example by taking the average of the three scores. As shown in Table III, overall, we can see that our model achieves good results even compared to the ground truth and outperforms the Transformer baseline. Interestingly, we observe that the Transformer baseline gets high syntactic and semantic scores, but very poor relevant scores. After manually examining some generated questions, we noticed that it generates many fluent and meaningful questions that are by no means relevant to the given KG subgraph. However, our model is able to generate more relevant questions possibly by better capturing the KG semantics and the answer.

E. Ablation Study

As shown in Table IV, we perform an ablation study to assess the performance impacts of different model components. 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.

F. Model Analysis

1) Effect of Node/Edge Embedding Initialization: We empirically compare two different ways of initializing node/edge embeddings when applying the Graph2Seq model.

TABLE II

AUTOMATIC EVALUATION RESULTS ON WQ AND PQ. THE METHODS MARKED WITH † ARE LARGE-SCALE PRETRAINED KG-TO-TEXT MODELS FINETUNED ON QG DATA WHILE OTHER METHODS DO NOT HAVE ACCESS TO SUCH PRETRAINING DATA. THE RESULTS MARKED IN BOLD AND WITH * INDICATE THE BEST AND SECOND BEST RESULTS, RESPECTIVELY

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
JointGT (T5)†	28.95	31.29*	54.47	60.45	45.38*	77.59
JointGT (BART)†	**30.02**	**32.05**	**55.60**	**65.89**	**48.25**	**78.87**
G2S+AE (ours)	29.45*	30.96	55.45*	61.48*	44.57	77.72*
$G2S_{edge}$ +AE (ours)	29.40	31.12	55.23	59.59	44.70	75.20

TABLE III

HUMAN EVALUATION RESULTS (±STANDARD DEVIATION) ON THE WQ TEST SET. THE RATING SCALE IS FROM 1 TO 5 (HIGHER SCORES INDICATE BETTER RESULTS)

Method	Syntactic	Semantic	Relevant	Overall
Transformer	**4.53 (0.18)**	**4.58 (0.22)**	2.65 (0.57)	3.92 (0.24)
G2S+AE	4.18 (0.30)	4.30 (0.27)	4.26 (0.34)	4.25 (0.26)
Ground-truth	4.30 (0.15)	4.50 (0.18)	**4.32 (0.32)**	**4.38 (0.19)**

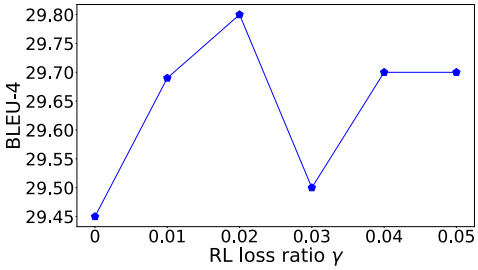


Fig. 3. Effect of RL ratio for G2S + AE + RL on WQ.

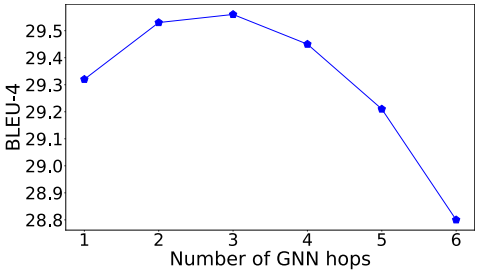


Fig. 4. Effect of number of GNN hops for G2S + AE on WQ.

As shown in Table V, encoding nodes and edges based on word embeddings of their textual attributes works better than based on their KG embeddings. This might be because it is difficult for an 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 NL 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.

2) Impact of Directionality on GNN Encoders: As shown in Table VI, we compare the performance of bidirectional Graph2Seq with unidirectional (i.e., forward and backward) Graph2Seq. We observe that utilizing the edge direction

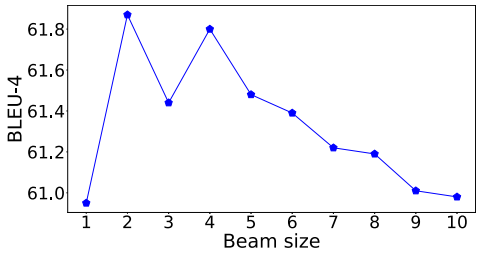


Fig. 5. Effect of beam search size for G2S + AE on PQ.

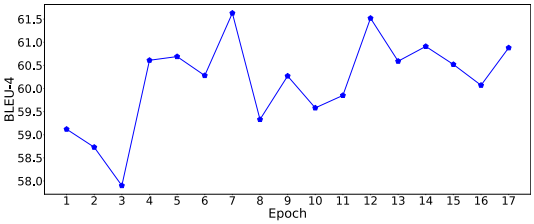


Fig. 6. Convergence analysis for G2S + AE on PQ.

information in the KG subgraph via bidirectional GNNs can significantly improve the model performance.

3) Results on the Two-Stage Training Strategy: Tables VII and VIII show the results of training our proposed G2S + AE model with a hybrid objective combining both cross-entropy loss and RL loss following the two-stage training strategy. We denote this variant as G2S + AE + RL. While the RL-based training strategy boosts the model performance on WQ, it does not help the model training on PQ. We suspect this is because the PQ dataset is easier compared to the WQ dataset, and therefore the benefit of RL-based finetuning on PQ is less significant. In order to study how the RL ratio γ affects the model performance, we report the test BLEU-4 scores on WQ corresponding to different values of γ, as shown in Fig. 3. As we can see, compared to $\gamma = 0$ which means no RL-based finetuning is applied, increasing the value of γ can help the model performance until a certain point.

4) Effect of the Number of GNN Hops: Fig. 4 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 some optimal value.

5) Effect of the Beam Search Size: Fig. 5 shows the impact of the beam size when applying beam search decoding during the testing phase on PQ. It indicates that beam search decoding significantly outperforms greedy search decoding (i.e., beam size = 1) and increasing the beam size can boost the model performance until some optimal value.

TABLE IV
ABLATION STUDY 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 V
EFFECT OF NODE/EDGE INIT. EMBEDDINGS ON WQ

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 VI
IMPACT OF DIRECTIONALITY FOR G2S + AE ON PQ

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 VII
RESULTS OF RL-BASED G2S + AE ON WQ

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

TABLE VIII
RESULTS OF RL-BASED G2S + AE ON PQ

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

TABLE IX
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

KG subgraph: (Egypt, administrative_divisions, Cairo), (Giza Necropolis, contained by, 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 ?

6) Convergence Analysis: Fig. 6 shows the changes in validation BLEU-4 scores over training epochs on PQ. As we can see, the model was able to converge quickly and achieved the best validation BLEU-4 score after epoch 7.

G. Case Study

As shown in Table IX, we conducted a 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. As we discussed in Section IV-F1, this might be because of the semantic gap between KG embeddings on the encoder side and word embeddings on the decoder side. Besides, with the node-level copying mechanism, the model was able to directly copy the entity name "giza necropolis" from the input KG subgraph into the output question. Last, incorporating the answer information helps generate more relevant and specific questions. For instance, given the target answer "Egypt," the model was able to produce a more specific question which is specifically asking for "what country" instead of "where."

TABLE X
ERROR ANALYSIS ON 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

KG subgraph: (martin luther king , jr ., speeches or presentations, /m/05r7ddy), (martin luther king , jr ., profession, writer), (martin luther king , jr ., profession, minister of religion), (martin luther king , jr ., profession, civil rights activist), (/m/05r7ddy, event, march on washington for jobs and freedom)
Gold: who was the speaker at march on washington for jobs and freedom facts ?
G2S+AE: who was the speaker in the march on washington for jobs and freedom ?
KG subgraph: (family guy, theme_song, family guy theme song), (family guy, regular cast, /m/02ntr0s), (/m/02ntr0s, actor, alex borstein), (/m/02ntr0s, character, lois griffin), (/m/02ntr0s, special performance type, voice)
Gold: who ' s the voice of stewie griffin from the tv program , with the family guy theme song ?
G2S+AE: who is the voice of the voice of the tv program with the family guy family guy theme song family guy theme song

H. Error Analysis

Table X shows some failure cases of our proposed G2S + AE model on the WQ test set. One common syntactic error pattern we observed is repeated words (e.g., repeated "the voice of" in the second example) in generated questions. Another error pattern is missing important pieces of information. For instance, in the second example,[2] our model failed to utilize the tuple (/m/02ntr0s, character, lois griffin) when generating the question. The coverage mechanism [77] is widely used in Seq2Seq models to encourage the full utilization of different tokens in the input text and penalize generating repetitive text. However, in our experiments, we found that applying the coverage mechanism did not help improve the overall evaluation scores. We conjecture that this might be

[2]In this example, the ground-truth question refers to the entity "stewie griffin" which is not included in the given input KG subgraph.

Fig. 7. Distribution of trigram prefixes of questions generated by G2S + AE on the WQ test set.

Fig. 8. Distribution of trigram prefixes of golden questions in the WQ test set.

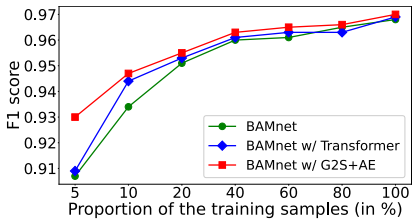


Fig. 9. Performance of QG-driven KBQA baseline under different proportions of training data.

because the coverage mechanism can also be too aggressive by encouraging the model to utilize irrelevant tuples in the input KG subgraph.

I. Visualization of the Generated Questions

Figs. 7 and 8 show the distributions of frequent trigram prefixes (i.e., frequency less than 5 not included) of the generated questions and golden questions on the WQ test set. As we can see, our G2S + AE model was able to generate diverse questions which have a similar distribution of trigram prefixes in comparison with the golden questions.

J. QG-Driven Data Augmentation for QA

One of the most important applications of QG is to generate more training data for QA tasks. In this section, we use our proposed QG model to generate more questions for training KBQA methods. We use WQ as our KBQA benchmark and randomly split it into 40%/20%/40% (train/dev/test) examples. As for the KBQA baseline, we use the state-of-the-art KBQA model called BAMnet [9] which directly retrieves answers from a KG by mapping questions and candidate answers into a joint embedding space. In order to examine the effect of QG-driven data augmentation on the KBQA task, we compare the BAMnet baseline with its two data augmentation variants, namely, BAMnet w/Transformer and BAMnet w/G2S + AE. More specifically, the BAMnet baseline is trained only on the part (i.e., x% of the whole training data) where gold questions are available, while the other two variants are trained on the combination of the gold questions and the questions are automatically generated by two QG models. Each x% corresponds to a data point in Fig. 9. We vary the value of x% from 5% all the way to 100% to examine the effectiveness of the QG-based data augmentation for KBQA with different training sizes. Note that given the x% training data, we further randomly split it to 80%/20% (train/dev) for training a QG model.

As shown in Fig. 9, we gradually increase the proportion (i.e., x%) of the training data and report the F1 score performance of the above three KBQA model variants. Here, the F1 score measures the overlap between the predicted and ground-truth answer sets. The results show that both QG models consistently help improve the KBQA performance when varying x% training data, and the performance boost is the most significant when training data is scarce (i.e., 5%, 10%). Notably, our G2S + AE model consistently outperforms the Transformer model in improving the KBQA performance.

V. Conclusion

In this article, we introduced a novel bidirectional Graph2Seq model for the KG-QG task. A novel node-level copying mechanism was proposed to allow the directly copying of node attributes from the KG subgraph to the output question. We explored different ways of initializing node/edge embeddings and handling multirelational graphs. Our model outperforms existing methods by a significant margin on two benchmarks.

In our experiments, we observed that node/edge embedding initialization has a big impact on the overall model performance. We would like to explore more effective ways of initializing node/edge embeddings in the future. Besides, how to effectively utilize the answer information is critical for generating relevant and meaningful questions. In this work, we introduced simple markup vectors to indicate whether an entity is a target answer or not. We leave more effective ways of answer utilization as future work. It is also beneficial to design more effective mechanisms to penalize generating repetitive text and encourage fully utilizing important information in the input KG subgraph. Another interesting direction is to integrate the QG model with KG completion systems. We expect this can be extremely beneficial when the input KG is incomplete and can potentially lead to generating more interesting and diverse questions.

Acknowledgment

The authors thank the editors and reviewers for their constructive feedback.

REFERENCES

[1] X. Du, J. Shao, and C. Cardie, "Learning to ask: Neural question generation for reading comprehension," 2017, *arXiv:1705.00106*.

[2] L. Song, Z. Wang, W. Hamza, Y. Zhang, and D. Gildea, "Leveraging context information for natural question generation," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol., (Short Papers)*, vol. 2, 2018, pp. 569–574.

[3] Y. Chen, L. Wu, and M. J. Zaki, "Reinforcement learning based graph-to-sequence model for natural question generation," in *Proc. ICLR*, 2020.

[4] L. Pan, Y. Xie, Y. Feng, T.-S. Chua, and M.-Y. Kan, "Semantic graphs for generating deep questions," 2020, *arXiv:2004.12704*.

[5] Y. Li et al., "Visual question generation as dual task of visual question answering," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6116–6124.

[6] J. Bao et al., "Table-to-text: Describing table region with natural language," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 5020–5027.

[7] D. Seyler, M. Yahya, and K. Berberich, "Knowledge questions from knowledge graphs," in *Proc. ACM SIGIR Int. Conf. Theory Inf. Retr.*, Oct. 2017, pp. 11–18.

[8] D. Tang, N. Duan, T. Qin, Z. Yan, and M. Zhou, "Question answering and question generation as dual tasks," 2017, *arXiv:1706.02027*.

[9] Y. Chen, L. Wu, and M. J. Zaki, "Bidirectional attentive memory networks for question answering over knowledge bases," in *Proc. Conf. North*, 2019, pp. 1–11.

[10] Y. Chen, A. Subburathinam, C.-H. Chen, and M. J. Zaki, "Personalized food recommendation as constrained question answering over a large-scale food knowledge graph," in *Proc. 14th ACM Int. Conf. Web Search Data Mining*, Mar. 2021, pp. 544–552.

[11] M. Heilman and N. A. Smith, "Good question! statistical ranking for question generation," in *Proc. Hum. Lang. Technol., Annu. Conf. North Amer. Chapter Assoc. Comput. Linguistics*, 2010, pp. 609–617.

[12] N. Mostafazadeh, I. Misra, J. Devlin, M. Mitchell, X. He, and L. Vanderwende, "Generating natural questions about an image," 2016, *arXiv:1603.06059*.

[13] D. Seyler, M. Yahya, and K. Berberich, "Generating quiz questions from knowledge graphs," in *Proc. 24th Int. Conf. World Wide Web*, May 2015, pp. 113–114.

[14] L. Song and L. Zhao, "Question generation from a knowledge base with web exploration," 2016, *arXiv:1610.03807*.

[15] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[16] I. V. Serban et al., "Generating factoid questions with recurrent neural networks: The 30M Factoid question-answer corpus," 2016, *arXiv:1603.06807*.

[17] S. Reddy, D. Raghu, M. M. Khapra, and S. Joshi, "Generating natural language question-answer pairs from a knowledge graph using a RNN based question generation model," in *Proc. 15th Conf. Eur. Chapter Assoc. Comput. Linguistics, Long Papers*, vol. 1, 2017, pp. 376–385.

[18] H. Elsahar, C. Gravier, and F. Laforest, "Zero-shot question generation from knowledge graphs for unseen predicates and entity types," 2018, *arXiv:1802.06842*.

[19] C. Liu, K. Liu, S. He, Z. Nie, and J. Zhao, "Generating questions for knowledge bases via incorporating diversified contexts and answer-aware loss," in *Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP)*, 2019, pp. 2431–2441.

[20] Y. Hu, H. Yang, G. Zhou, and J. X. Huang, "Generating factoid questions with question type enhanced representation and attention-based copy mechanism," *ACM Trans. Asian Low-Resource Lang. Inf. Process.*, vol. 21, no. 2, pp. 1–18, Mar. 2022.

[21] I. Sutskever, O. Vinyals, and Q. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. NIPS*, 2014, pp. 3104–3112.

[22] K. Cho et al., "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1724–1734.

[23] V. Kumar, Y. Hua, G. Ramakrishnan, G. Qi, L. Gao, and Y.-F. Li, "Difficulty-controllable multi-hop question generation from knowledge graphs," in *Proc. Int. Semantic Web Conf.* Cham, Switzerland: Springer, 2019, pp. 382–398.

[24] S. Bi, X. Cheng, Y.-F. Li, Y. Wang, and G. Qi, "Knowledge-enriched, type-constrained and grammar-guided question generation over knowledge bases," in *Proc. 28th Int. Conf. Comput. Linguistics*, 2020, pp. 2776–2786.

[25] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[26] X. Shen, J. Chen, J. Chen, C. Zeng, and Y. Xiao, "Diversified query generation guided by knowledge graph," in *Proc. 15th ACM Int. Conf. Web Search Data Mining*, Feb. 2022, pp. 897–907.

[27] W. Chen, Y. Su, X. Yan, and W. Y. Wang, "KGPT: Knowledge-grounded pre-training for data-to-text generation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2020, pp. 8635–8648.

[28] P. Ke et al., "JointGT: Graph-text joint representation learning for text generation from knowledge graphs," in *Proc. Findings Assoc. Comput. Linguistics, (ACL-IJCNLP)*, 2021, pp. 2526–2538.

[29] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.

[30] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. 34th Int. Conf. Mach. Learning*, vol. 70, 2017, pp. 1263–1272.

[31] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1024–1034.

[32] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," 2015, *arXiv:1511.05493*.

[33] Y. Chen, L. Wu, and M. J. Zaki, "Iterative deep graph learning for graph neural networks: Better and robust node embeddings," in *Proc. NeurIPS*, 2020, pp. 19314–19326.

[34] N. Liu, X. Wang, L. Wu, Y. Chen, X. Guo, and C. Shi, "Compact graph structure learning via mutual information compression," in *Proc. ACM Web Conf.*, Apr. 2022, pp. 1601–1610.

[35] J. Bastings, I. Titov, W. Aziz, D. Marcheggiani, and K. Sima'an, "Graph convolutional encoders for syntax-aware neural machine translation," 2017, *arXiv:1704.04675*.

[36] L. Song, Y. Zhang, Z. Wang, and D. Gildea, "A graph-to-sequence model for AMR-to-text generation," 2018, *arXiv:1805.02473*.

[37] Y. Chen, L. Wu, and M. J. Zaki, "GraphFlow: Exploiting conversation flow with graph neural networks for conversational machine comprehension," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Jul. 2020, pp. 1230–1236.

[38] L. Wu et al., "Graph neural networks for natural language processing: A survey," *Found. Trends Mach. Learn.*, vol. 16, no. 2, pp. 119–328, 2023.

[39] L. Wu, Y. Chen, H. Ji, and B. Liu, "Deep learning on graphs for natural language processing," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2021, pp. 2651–2653.

[40] D. Beck, G. Haffari, and T. Cohn, "Graph-to-sequence learning using gated graph neural networks," 2018, *arXiv:1806.09835*.

[41] K. Xu, L. Wu, Z. Wang, M. Yu, L. Chen, and V. Sheinin, "Exploiting rich syntactic information for semantic parsing with graph-to-sequence model," 2018, *arXiv:1808.07624*.

[42] S. Liu, Y. Chen, X. Xie, J. K. Siow, and Y. Liu, "Retrieval-augmented generation for code summarization via hybrid GNN," in *Proc. ICLR*, 2021.

[43] K. Xu, L. Wu, Z. Wang, Y. Feng, M. Witbrock, and V. Sheinin, "Graph2Seq: Graph to sequence learning with attention-based neural networks," 2018, *arXiv:1804.00823*.

[44] K. Xu, L. Wu, Z. Wang, Y. Feng, and V. Sheinin, "SQL-to-text generation with graph-to-sequence model," 2018, *arXiv:1809.05255*.

[45] D. Marcheggiani and L. Perez-Beltrachini, "Deep graph convolutional encoders for structured data to text generation," 2018, *arXiv:1810.09995*.

[46] P. Vougiouklis et al., "Neural wikipedian: Generating textual summaries from knowledge base triples," *J. Web Semantics*, vols. 52–53, pp. 1–15, Oct. 2018.

[47] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 2787–2795.

[48] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1532–1543.

[49] S. Haussmann et al., "FoodKG: A semantics-driven knowledge graph for food recommendation," in *Proc. Int. Semantic Web Conf.* Cham, Switzerland: Springer, 2019, pp. 146–162.

[50] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," 2017, *arXiv:1710.10903*.

[51] L. F. R. Ribeiro, C. Gardent, and I. Gurevych, "Enhancing AMR-to-text generation with dual graph representations," 2019, *arXiv:1909.00352*.

[52] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 3693–3702.

[53] F. W. Levi, *Finite Geometrical Systems: Six Public Lectues Delivered in February, 1940, at the University of Calcutta*. Kolkata, India: Univ. Calcutta, 1942.

[54] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 807–814.

[55] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, *arXiv:1409.0473*.

[56] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," 2015, *arXiv:1508.04025*.

[57] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," 2017, *arXiv:1704.04368*.

[58] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2692–2700.

[59] J. Gu, Z. Lu, H. Li, and V. O. K. Li, "Incorporating copying mechanism in Sequence-to-Sequence learning," 2016, *arXiv:1603.06393*.

[60] R. Koncel-Kedziorski, D. Bekal, Y. Luan, M. Lapata, and H. Hajishirzi, "Text generation from knowledge graphs with graph transformers," 2019, *arXiv:1904.02342*.

[61] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1171–1179.

[62] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, "Sequence level training with recurrent neural networks," 2015, *arXiv:1511.06732*.

[63] Y. Wu et al., "Google's neural machine translation system: Bridging the gap between human and machine translation," 2016, *arXiv:1609.08144*.

[64] R. Paulus, C. Xiong, and R. Socher, "A deep reinforced model for abstractive summarization," 2017, *arXiv:1705.04304*.

[65] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 229–256, 1992.

[66] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel, "Self-critical sequence training for image captioning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 7008–7024.

[67] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush, "OpenNMT: Open-source toolkit for neural machine translation," 2017, *arXiv:1701.02810*.

[68] Google. (2018). *Freebase Data Dumps*. [Online]. Available: https://developers.google.com/freebase

[69] W.-T. Yih, M. Richardson, C. Meek, M.-W. Chang, and J. Suh, "The value of semantic parse labeling for knowledge base question answering," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics (Short Papers)*, vol. 2, 2016, pp. 201–206.

[70] A. Talmor and J. Berant, "The web as a knowledge-base for answering complex questions," 2018, *arXiv:1803.06643*.

[71] M. Zhou, M. Huang, and X. Zhu, "An interpretable reasoning network for multi-relation question answering," 2018, *arXiv:1801.04726*.

[72] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: A method for automatic evaluation of machine translation," in *Proc. 40th Annu. Meeting Assoc. Comput. Linguistics (ACL)*, 2001, pp. 311–318.

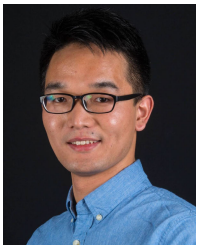
[73] A. Lavie and A. Agarwal, "Meteor: An automatic metric for MT evaluation with high levels of correlation with human judgments," in *Proc. 2nd Workshop Stat. Mach. Transl. (StatMT)*, 2007, pp. 65–72.

[74] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, 2004, pp. 74–81.

[75] D. P. Kingma, T. Salimans, and M. Welling, "Variational dropout and the local reparameterization trick," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2575–2583.

[76] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

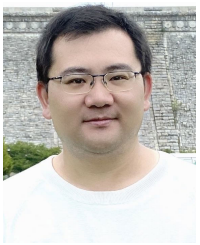
[77] Z. Tu, Z. Lu, Y. Liu, X. Liu, and H. Li, "Modeling coverage for neural machine translation," 2016, *arXiv:1601.04811*.



Yu Chen received the Ph.D. degree in computer science from the Rensselaer Polytechnic Institute, Troy, NY, USA, in 2020.

He is currently a Senior Research Scientist with Meta AI, Menlo Park, CA, USA. His work has been published at top-ranked conferences including, but not limited to, NeurIPS, ICML, ICLR, AAAI, IJCAI, NAACL, KDD, WSDM, ISWC, and AMIA. He was one of the book chapter contributors of the book *Graph Neural Networks: Foundations, Frontiers, and Applications*. He delivered a series of DLG4NLP Tutorials at NAACL'21, SIGIR'21, KDD'21, IJCAI'21, AAAI'22, and TheWebConf'22. His work has been covered in popular technology and marketing publications including World Economic Forum, TechXplore, TechCrunch, Ad Age, and Adweek. He is a Co-Inventor of four filed US patents. His research interests lie at the intersection of machine learning (deep learning) and natural language processing, with a particular emphasis on the fast-growing field of graph neural networks and their applications in various domains.

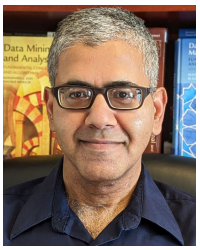
Dr. Chen was a recipient of the Best Student Paper Award of AAAI DLGMA'20. He has served as a PC Member/Reviewer for various conferences (e.g., ACL, EMNLP, NAACL, EACL, AAAI, IJCAI, KDD, CIKM, and LoG) and journals (e.g., the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE (TPAMI), the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS (TNNLS), *IJIS*, the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING (TKDE), *TKDD*, and *DAMI*).



Lingfei Wu (Member, IEEE) received the Ph.D. degree in computer science from the College of William and Mary, Williamsburg, VA, USA, in 2016.

He is currently an Engineering Manager with Pinterest, San Francisco, CA, USA. He has published one book (in GNNs) and more than 100 top-ranked conference and journal papers and is a co-inventor of more than 40 filed US patents. Because of the high commercial value of his patents, he received eight invention achievement awards and was appointed as IBM Master Inventor, class of 2020.

Dr. Wu was a recipient of the Best Paper Award and Best Student Paper Award at several conferences such as IEEE ICC'19, AAAI workshop on DLGMA'20, and KDD workshop on DLG'19. His research has been featured in numerous media outlets, including NatureNews, YahooNews, AP News, PR Newswire, The Time Weekly, Venturebeat, MIT News, and SIAM News. He has served as Industry and Government Program Co-Chairs of IEEE BigData'22, Sponsorship Co-Chairs of KDD'22, and Associate Conference Co-Chairs of AAAI'21, and is the founding co-chair for several workshops such as Deep Learning on Graphs (with AAAI'20-22 and KDD'19-22). He has also served as an Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS and *ACM Transactions on Knowledge Discovery from Data*.



Mohammed J. Zaki (Fellow, IEEE) received the Ph.D. degree in computer science from the University of Rochester, Rochester, NY, USA, in 1998.

He is currently a Professor and the Department Head of computer science with RPI. He has around 300 publications (and six patents), including the *Data Mining and Machine Learning* textbook. His research interests focus on novel data mining and machine-learning techniques.

Dr. Zaki is a fellow of the ACM and AAAS. He was a recipient of the NSF CAREER Award and the Department of Energy Early Career Principal Investigator Award, as well as the HP Innovation Research Award, and Google Faculty Research Award. His research is supported in part by NSF, DARPA, NIH, DOE, IBM, Google, HP, and Nvidia. He was the Program Co-Chair for SDM'08, SIGKDD'09, PAKDD'10, BIBM'11, CIKM'12, ICDM'12, IEEE BigData'15, CIKM'18, and CIKM'22. He is currently serving on the Board of Directors for ACM SIGKDD. He is the Founding Co-Chair of the BIOKDD series of workshops. He is currently an Associate Editor of *Data Mining and Knowledge Discovery*. He has also served as an Area Editor for *Statistical Analysis and Data Mining* and as an Associate Editor for *ACM TKDD* and *SNAM*.