JointGT: Graph-Text Joint Representation Learning for Text Generation from Knowledge Graphs

Pei Ke[1], **Haozhe Ji**[1], **Yu Ran**[2], **Xin Cui**[2], **Liwei Wang**[3], **Linfeng Song**[4], **Xiaoyan Zhu**[1], **Minlie Huang**[1]*

[1]The CoAI group, Department of Computer Science and Technology,
Institute for Artificial Intelligence, State Key Lab of Intelligent Technology and Systems,
Beijing National Research Center for Information Science and Technology,
Tsinghua University, Beijing 100084, China

[2]Sogou Inc., Beijing, China [3]The Chinese University of Hong Kong [4]Tencent AI Lab

{kp17,jhz20}@mails.tsinghua.edu.cn, {zxy-dcs,aihuang}@tsinghua.edu.cn

Abstract

Existing pre-trained models for knowledge-graph-to-text (KG-to-text) generation simply fine-tune text-to-text pre-trained models such as BART or T5 on KG-to-text datasets, which largely ignore the graph structure during encoding and lack elaborate pre-training tasks to explicitly model graph-text alignments. To tackle these problems, we propose a graph-text joint representation learning model called JointGT. During encoding, we devise a structure-aware semantic aggregation module which is plugged into each Transformer layer to preserve the graph structure. Furthermore, we propose three new pre-training tasks to explicitly enhance the graph-text alignment including respective text / graph reconstruction, and graph-text alignment in the embedding space via Optimal Transport. Experiments show that JointGT obtains new state-of-the-art performance on various KG-to-text datasets[1].

1 Introduction

Knowledge-graph-to-text (KG-to-text) generation aims to generate high-quality texts which are consistent with input graphs (Gardent et al., 2017). This task requires to simultaneously encode the graph structure and the content, and effectively leverage the input graphs in the decoding process (Zhao et al., 2020). As a major natural language generation (NLG) task that connects knowledge graphs and texts, this task can further promote the applicability of knowledge graphs in more realistic NLG scenarios, such as knowledge-grounded dialogue generation (Zhou et al., 2018a) and story generation (Guan et al., 2019; Ji et al., 2020).

Due to the limited amount of graph-text parallel data, it's hard for typical neural text generation models to learn the alignments between source entities / relations and target tokens from scratch (Guo et al., 2020; Fu et al., 2020). Recent work resorts to constructing general-purpose pre-trained language models for KG-to-text generation. The most common and simple way is to linearize input graphs into text sequences, and directly fine-tune text-to-text Transformer-based pre-trained models like GPT (Radford et al., 2018, 2019), BART (Lewis et al., 2020) or T5 (Raffel et al., 2020) on KG-to-text datasets (Ribeiro et al., 2020a; Kale and Rastogi, 2020). Benefiting from self-supervised pre-training on large-scale unlabelled text corpora, pre-trained language models can generate high-quality texts via simply fine-tuning, and outperform other models with sophisticated structures.

Despite the superior performance of fine-tuning pre-trained models on KG-to-text datasets, we argue that building pre-trained models for KG-to-text generation still faces two major challenges: 1) **Structural information loss during encoding**. Most of the existing pre-trained models capture contextual information via bidirectional Transformers (Devlin et al., 2019), which include full attention connections. This model structure may neglect the structural information when encoding knowledge graphs since the relation between each pair of input entities is not explicitly considered (Zhu et al., 2019). 2) **Absence of explicit graph-text alignments**. Existing work on pre-trained models for text generation commonly adopts auto-encoding or auto-regressive text reconstruction to learn text-text alignments, which encodes the corrupted text sequence and decodes the original sequence (Lewis et al., 2020; Raffel et al., 2020). Since knowledge graphs may possess more complex structures than text sequences, it's hard to explicitly learn graph-text alignments by directly using the pre-training tasks based on text reconstruction.

Thus, we propose a graph-text joint represen-

* Corresponding author

[1]The data, codes, and model parameters are available at https://github.com/thu-coai/JointGT.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2526–2538
August 1–6, 2021. ©2021 Association for Computational Linguistics

tation learning framework called *JointGT* to deal with the above challenges. **Firstly**, to alleviate the structural information loss during encoding, we devise a simple structure-aware semantic aggregation module at each Transformer layer to aggregate contextual information following the graph structure. **Secondly**, we propose three pre-training tasks including graph enhanced text reconstruction, text enhanced graph reconstruction, and graph-text embedding alignment to explicitly build the connection between knowledge graphs and text sequences. The first two tasks are expected to enhance the graph-text alignment in the discrete vocabulary space, where our model is required to predict the masked information of graphs / texts based on the observed information of texts / graphs. And the third task is designed to model the graph-text alignment in the continuous embedding space via Optimal Transport (Peyré and Cuturi, 2019) to match the hidden representations of graphs and texts. Our contributions are as follows:

- We propose a novel pre-trained model called JointGT for KG-to-text generation tasks. This model adopts a structure-aware semantic aggregation module to model the structure of an input graph at each Transformer layer, and utilizes three pre-training tasks to explicitly learn graph-text alignments in the discrete and continuous spaces.
- We conduct experiments on the datasets of KG-to-text generation including WebNLG, WebQuestions and PathQuestions. Results show that JointGT achieves new state-of-the-art performance on KG-to-text generation.

2 Related Work

KG-to-Text Generation

Recent studies on KG-to-text generation tasks mainly fall into three aspects: 1) *Encoder modification*: To alleviate the structural information loss of sequence encoders with the input of linearized graphs (Gardent et al., 2017; Trisedya et al., 2018; Moryossef et al., 2019), researchers focus on more complex encoder structures for better graph representations, such as graph neural networks (Marcheggiani and Perez-Beltrachini, 2018; Ribeiro et al., 2020b) and graph Transformers (Koncel-Kedziorski et al., 2019; Schmitt et al., 2020a). 2) *Unsupervised training*: researchers devise unsupervised training objectives to jointly learn the tasks of graph-to-text and text-to-graph conversion with non-parallel graph-text data (Schmitt et al., 2020b; Guo et al., 2020; Jin et al., 2020). 3) *Building pre-trained models*: With the development of pre-trained NLG models such as GPT (Radford et al., 2018, 2019), BART (Lewis et al., 2020) and T5 (Raffel et al., 2020), recent work directly fine-tunes these models on graph-to-text datasets and reports impressive performance (Ribeiro et al., 2020a; Kale and Rastogi, 2020; Chen et al., 2020b; Mager et al., 2020).

Compared with the existing work on pre-trained models for KG-to-text generation, our model utilizes pre-training methods to explicitly learn graph-text alignments instead of directly fine-tuning text-to-text pre-trained models on KG-to-text datasets.

KG-Enhanced Pre-Trained Models

Another line of related studies is pre-trained models enhanced by knowledge graphs for natural language understanding (NLU). The motivation of these models is to incorporate knowledge graphs into pre-trained models to facilitate the understanding of entities and relations in natural language. Early work including ERNIE (Zhang et al., 2019) and KnowBERT (Peters et al., 2019) directly uses fixed entity embeddings based on TransE (Bordes et al., 2013) or word vectors (Mikolov et al., 2013) during pre-training. Recent work like KEPLER (Wang et al., 2021) and JAKET (Yu et al., 2020) resorts to jointly pre-training graph-text representations. Specifically, they encode the textual descriptions of entities with pre-trained language models as entity embeddings and jointly optimize the knowledge embedding objective and the masked language modeling objective.

In comparison, our model focuses on joint pre-training methods on knowledge graph encoding and sequence decoding in KG-to-text generation tasks, rather than considering graph-text joint encoding methods in NLU tasks.

3 Method

3.1 Task Definition and Model Overview

Given a knowledge graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{e_1, e_2, \cdots, e_{|\mathcal{V}|}\}$ denotes the entity set and $\mathcal{E} = (r_{ij})_{|\mathcal{V}|\times|\mathcal{V}|}$ indicates the relations connecting the entities, and its linearized version $\mathcal{G}_{linear} = (w_1, w_2, \cdots, w_m)$ which consists of m tokens, our goal is to generate a text sequence $X = (x_1, x_2, \cdots, x_n)$ which is consistent with the input graph.

Our model is built on pre-trained encoder-decoder models like BART (Lewis et al., 2020) and T5 (Raffel et al., 2020). First of all, we follow the existing work (Chen et al., 2020b) to linearize knowledge graphs in the form of triple lists (as shown in Figure 1), and devise a simple structure-aware semantic aggregation module which is plugged into each Transformer layer of the encoder to preserve the structural information of input graphs (§3.2). Then, we propose three pre-training tasks including graph / text reconstruction in the discrete vocabulary space and graph-text matching in the continuous embedding space, which enable our model to jointly learn the representations of knowledge graphs and texts (§3.3).

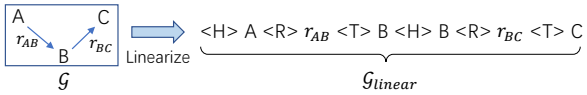


Figure 1: Illustration of linearizing knowledge graphs into text sequences. The special tokens <H>, <R> and <T> mean the head entity, relation and tail entity in the knowledge triples, respectively.

3.2 Model Structure

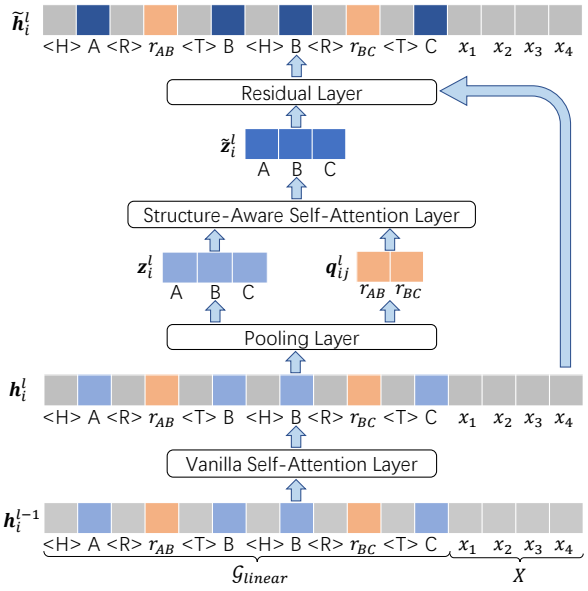


Figure 2: Structure-aware semantic aggregation module at each layer of the Transformer encoder. This module contains a pooling layer to obtain the contextual semantic representations of entities ($\boldsymbol{z}_i^l$) and relations ($\boldsymbol{q}_{ij}^l$) from the output of the vanilla self-attention layer ($\boldsymbol{h}_i^l$), a structure-aware self-attention layer to aggregate the entity representations ($\tilde{\boldsymbol{z}}_i^l$) based on the graph structure, and a residual layer to fuse the contextual and structural representations ($\tilde{\boldsymbol{h}}_i^l$).

To simultaneously leverage the contextual representation from pre-trained models and preserve the structural information, we devise a structure-aware semantic aggregation module in the Transformer encoder. Assume that the input of our encoder during pre-training is the linearized graph $\mathcal{G}_{linear}$ and the corresponding text sequence X (which may be corrupted or empty in some pre-training tasks), the self-attention layer in the l-th Transformer layer can be formulated as follows[2]:

$$\begin{aligned}
\boldsymbol{h}_i^l &= \sum_{j=1}^{m+n} \alpha_{ij}^l (\boldsymbol{h}_j^{l-1} \boldsymbol{W}^V) \\
\alpha_{ij}^l &= \frac{\exp(t_{ij}^l)}{\sum_{p=1}^{m+n} \exp(t_{ip}^l)} \\
t_{ij}^l &= \frac{\left(\boldsymbol{h}_i^{l-1} \boldsymbol{W}^Q\right)\left(\boldsymbol{h}_j^{l-1} \boldsymbol{W}^K\right)^\top}{\sqrt{d_k}} \\
& i = 1, 2, \cdots, m+n
\end{aligned} \tag{1}$$

where $\boldsymbol{W}^Q, \boldsymbol{W}^K, \boldsymbol{W}^V$ are the model parameters and d_k denotes the dimension of query / key / value vectors. The fully-connected attention captures rich contextual semantic relationship among the entities, relations and the tokens of text sequences, but is not sufficient to encode the structural information of input graphs. Thus, we devise a structure-aware semantic aggregation module on top of vanilla self-attention, as shown in Figure 2. First of all, we utilize a mean pooling layer[3] to obtain the representation of each entity and relation from the output of the vanilla self-attention layer:

$$\begin{aligned}
\boldsymbol{z}_i^l &= \text{pooling}(\{\boldsymbol{h}_p^l | p \in \mathcal{P}(e_i), 1 \le p \le m\}) \\
\boldsymbol{q}_{ij}^l &= \text{pooling}(\{\boldsymbol{h}_p^l | p \in \mathcal{P}(r_{ij}), 1 \le p \le m\}) \\
& i = 1, \cdots, |\mathcal{V}|; \quad j = 1, \cdots, |\mathcal{V}|
\end{aligned} \tag{2}$$

where $\mathcal{P}(e_i)/\mathcal{P}(r_{ij})$ means the set of positions occupied by e_i / r_{ij} in the linearized graph. Note that $\boldsymbol{q}_{ij}^l$ will be set to an all-zero vector if there is no relation between e_i and e_j. Then we update entity representations with a structure-aware self-

[2]We take a single attention head as an example in this section. In practice, we use our proposed method in the multi-head attention.

[3]We find that there is no significant difference in the model performance between mean pooling and other aggregation functions like max pooling.

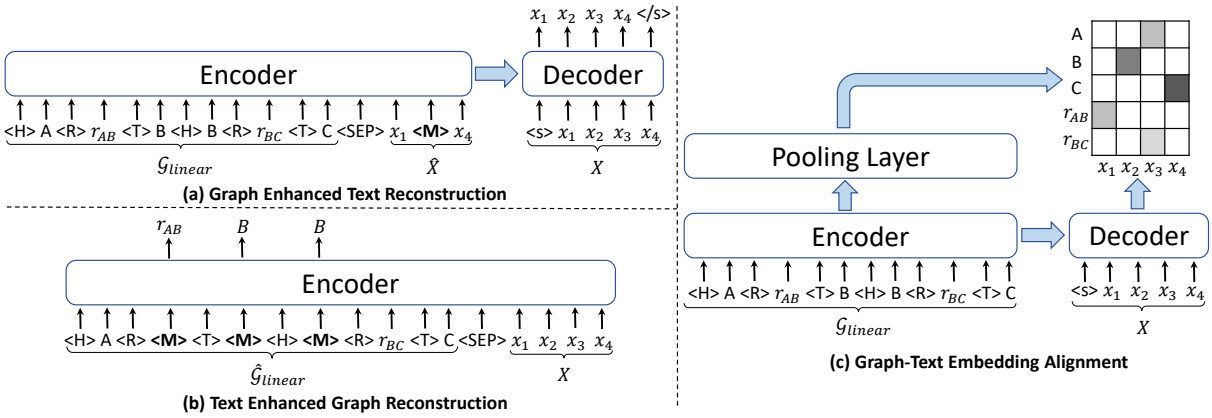


Figure 3: Overview of our proposed pre-training tasks: (a) Graph enhanced text reconstruction: reconstructing the text sequence given the complete graph. (b) Text enhanced graph reconstruction: predicting the masked entities and relations of the corrupted graph conditioned on the complete text. (c) Graph-text embedding alignment: matching the embedding vectors of the knowledge graph and the text via Optimal Transport. The special token <SEP> is to separate the linearized graph and the text, while <M> denotes the placeholder for masked tokens.

attention layer (Shaw et al., 2018):

$$\tilde{\boldsymbol{z}}_i^l = \sum_{j=1}^{|\mathcal{V}|} \beta_{ij}^l (\boldsymbol{z}_j^l \boldsymbol{W}^{VS} + \boldsymbol{q}_{ij}^l \boldsymbol{W}^{VR})$$
$$\beta_{ij}^l = \frac{\exp(u_{ij}^l)}{\sum_{p=1}^{|\mathcal{V}|} \exp(u_{ip}^l)} \quad (3)$$

$$u_{ij}^l = \frac{\left(\boldsymbol{z}_i^l \boldsymbol{W}^{QS}\right)\left(\boldsymbol{z}_j^l \boldsymbol{W}^{KS} + \boldsymbol{q}_{ij}^l \boldsymbol{W}^{KR}\right)^\top}{\sqrt{d_k}}$$
$$i = 1, 2, \cdots, |\mathcal{V}|$$

where $\boldsymbol{W}^{QS}, \boldsymbol{W}^{KS}, \boldsymbol{W}^{VS}, \boldsymbol{W}^{KR}, \boldsymbol{W}^{VR}$ are the weight matrices in the structure-aware self-attention. This layer integrates the contextual semantic representation of entities and relations based on the graph structure, thereby injecting the structural information into the vanilla Transformer layer. Finally, we use a residual layer to fuse semantic and structural representations of entities, and obtain the hidden states for the following computation:

$$\tilde{\boldsymbol{h}}_i^l = \begin{cases} \boldsymbol{h}_i^l + \tilde{\boldsymbol{z}}_j^l, & i \in \mathcal{P}(e_j) \\ \boldsymbol{h}_i^l, & otherwise. \end{cases} \quad (4)$$
$$i = 1, \cdots, m+n; \quad j = 1, \cdots, |\mathcal{V}|$$

Compared with existing structure-aware Transformer encoders (Zhu et al., 2019; Song et al., 2020) that either use the entity and relation embeddings from an external knowledge embedding model or directly learn them as model parameters, our encoder obtains the entity and relation embeddings via contextual semantic representations. This design fully employs the effective contextual representations from the existing pre-trained models while preserving the structural information, and enables our model to generalize to new entities and relations better when fine-tuned to the datasets with a different knowledge graph.

3.3 Pre-Training Task

Given the input graph $\mathcal{G}$ and its corresponding text sequence X, the goal of our pre-training task is to jointly learn the graph encoder and sequence decoder to enhance graph-text alignments, which can benefit the downstream tasks of KG-to-text generation. We devise three pre-training tasks to explicitly learn graph-text alignments in both discrete and continuous spaces.

3.3.1 Graph Enhanced Text Reconstruction

The purpose of graph enhanced text reconstruction is to recover the masked text sequence based on the complete knowledge graph, as shown in Figure 3. Assume that $\hat{X}$ denotes the masked text sequence, we can formulate the loss function of this pre-training task as follows:

$$\mathcal{L}_{text} = -\log P(X|\mathcal{G}, \hat{X})$$
$$= -\sum_{i=1}^{n} \log P(x_i|\mathcal{G}, \hat{X}, x_{<i}) \quad (5)$$

To construct $\hat{X}$, we masked the entity words with a probability of 40% and other words with 20% since entity words are more important in the

task of KG-to-text generation. We also follow the existing work (Lewis et al., 2020) to merge the consecutive mask tokens into one mask token to increase the difficulty of text reconstruction. This task enables our model to utilize the knowledge graph to reconstruct the corrupted text sequence, which explores the connection between them in the discrete vocabulary space.

3.3.2 Text Enhanced Graph Reconstruction

As shown in Figure 3, this pre-training task aims to recover the corrupted graph according to the information of the text sequence. Given the corrupted knowledge graph $\hat{\mathcal{G}}$ with masked entities and relations, and the complete text sequence X, the loss function is to recover the masked entities and relations in the linearized knowledge graph:

$$\begin{aligned}\mathcal{L}_{graph} &= -\log P(\mathcal{G}|\hat{\mathcal{G}}, X) \\ &= -\sum_{i=1}^{m} M_i \log P(w_i|\hat{\mathcal{G}}, X) \end{aligned} \quad (6)$$

where M_i denotes an indicator function and equals 1 if and only if w_i is masked. We empirically set the masking probability of entities / relations as 40% / 20%. This task explicitly exerts the impact of the text on the graph reconstruction, thereby guiding the encoder to focus more on the entities and relations that may appear in the text.

3.3.3 Graph-Text Embedding Alignment

This pre-training task is devised to encourage the graph-text alignment in the embedding space. We use Optimal Transport (OT), which is commonly used in the cross-domain alignment (Chen et al., 2020a), to calculate the minimum cost of transporting the graph representation from the encoder to the text representation from the decoder (and vice versa). As shown in Figure 3, the input of the encoder is the linearized knowledge graph $\mathcal{G}_{linear}$ while the input of the decoder is the text sequence X. Assume that $\boldsymbol{H}^L = (\boldsymbol{h}_1^L, \boldsymbol{h}_2^L, \cdots, \boldsymbol{h}_m^L)$ indicates the final hidden states of the encoder, we can similarly acquire the entity and relation representations via mean pooling:

$$\begin{aligned}\boldsymbol{z}_i^L =& \text{pooling}(\{\boldsymbol{h}_p^L | p \in \mathcal{P}(e_i), 1 \le p \le m\}) \\ \boldsymbol{q}_{ij}^L =& \text{pooling}(\{\boldsymbol{h}_p^L | p \in \mathcal{P}(r_{ij}), 1 \le p \le m\}) \\ & i = 1, \cdots, |\mathcal{V}|; \quad j = 1, \cdots, |\mathcal{V}| \end{aligned} \quad (7)$$

Let $\mathcal{G}_{seq} = \mathcal{V} \cup \mathcal{E} = (g_1, g_2, \cdots, g_{|\mathcal{V}|+|\mathcal{E}|})$ denotes the sequence of all the entities and relations in $\mathcal{G}$, we can directly obtain the contextual embedding vectors $\boldsymbol{H}^{\mathcal{G}} = (\boldsymbol{h}_1^{\mathcal{G}}, \cdots, \boldsymbol{h}_{|\mathcal{V}|+|\mathcal{E}|}^{\mathcal{G}})$ for each entity and relation from Equation 7. We can also acquire the embedding vectors of X from the decoder's final hidden states, which is denoted by $\boldsymbol{S} = (\boldsymbol{s}_1, \boldsymbol{s}_2, \cdots, \boldsymbol{s}_n)$.

To model the alignment between graphs and texts in the embedding space, we regard $\mathcal{G}_{seq}$ as a discrete distribution $\boldsymbol{\mu} = \sum_{i=1}^{|\mathcal{V}|+|\mathcal{E}|} \boldsymbol{a}_i \delta_{g_i}$ and X as $\boldsymbol{\upsilon} = \sum_{j=1}^{n} \boldsymbol{b}_j \delta_{x_j}$, where $\boldsymbol{a} = \{\boldsymbol{a}_i\}_{i=1}^{|\mathcal{V}|+|\mathcal{E}|}$ and $\boldsymbol{b} = \{\boldsymbol{b}_j\}_{j=1}^{n}$ satisfy $\sum_{i=1}^{|\mathcal{V}|+|\mathcal{E}|} \boldsymbol{a}_i = \sum_{j=1}^{n} \boldsymbol{b}_j = 1$, and δ_{g_i} / δ_{x_j} indicates the Dirac function centered on g_i / x_j. Then, we utilize the OT distance between $\boldsymbol{\mu}$ and $\boldsymbol{\upsilon}$ as the loss function, which is defined as the solution of the following problem:

$$\mathcal{L}_{OT} = \min_{\boldsymbol{T} \in \Pi(\boldsymbol{a}, \boldsymbol{b})} \sum_{i=1}^{|\mathcal{V}|+|\mathcal{E}|} \sum_{j=1}^{n} \boldsymbol{T}_{ij} \cdot d(g_i, x_j) \quad (8)$$

$$\Pi(\boldsymbol{a}, \boldsymbol{b}) = \{\boldsymbol{T} \in \mathbb{R}_+^{(|\mathcal{V}|+|\mathcal{E}|)\times n} | \boldsymbol{T} \cdot \mathbf{1}_n = \boldsymbol{a}, \boldsymbol{T}^\top \cdot \mathbf{1}_{|\mathcal{V}|+|\mathcal{E}|} = \boldsymbol{b}\}$$

where $\boldsymbol{T}$ denotes a transport plan, $\mathbf{1}_{|\mathcal{V}|+|\mathcal{E}|}$ / $\mathbf{1}_n$ indicates the $(|\mathcal{V}|+|\mathcal{E}|)$ / n -dimensional all-one vector respectively, and $d(g_i, x_j)$ is the cost function of transporting g_i to x_j. We follow the existing work (Chen et al., 2020c) to adopt the cosine distance between the contextual embedding vectors of g_i and x_j as the cost function, which is defined as $d(g_i, x_j) = 1 - \frac{\boldsymbol{h}_i^{\mathcal{G}} \boldsymbol{s}_j}{\|\boldsymbol{h}_i^{\mathcal{G}}\|_2 \|\boldsymbol{s}_j\|_2}$. Since the exact minimization over $\boldsymbol{T}$ is computationally intractable, we utilize IPOT algorithm (Xie et al., 2019) to approximate the OT distance and iteratively obtain the solution of $\boldsymbol{T}$ (more details are provided in the Appendix A). After solving $\boldsymbol{T}$, $\mathcal{L}_{OT}$ can serve as an alignment loss to optimize the model parameters. This task builds the connection between the contextual embedding vectors of knowledge graphs and texts, and explicitly promotes the graph-text alignment in the continuous space.

4 Experiment

4.1 Pre-training Dataset and Implementation

We used KGTEXT (Chen et al., 2020b) as our pre-training dataset. This dataset contains 7M graph-text data pairs, where texts are crawled from English Wikidump[4] and the corresponding knowledge graphs are acquired by querying WikiData with the

[4] https://dumps.wikimedia.org

Dataset	#Param	WebNLG(U)			WebNLG(C)			WebQuestions			PathQuestions		
Model		BLEU	METEOR	ROUGE	BLEU	METEOR	ROUGE	BLEU	METEOR	ROUGE	BLEU	METEOR	ROUGE
SOTA-NPT	-	61.00†	42.00†	71.00†	48.00†	36.00†	65.00†	29.45‡	30.96‡	55.45‡	61.48‡	44.57‡	77.72‡
KGPT	177M	64.11♯	46.30♯	74.57♯	-	-	-	-	-	-	-	-	-
BART	140M	64.55	46.51	75.13	56.65	44.51	70.94	29.61	31.48	55.42	63.74	47.23	77.76
T5	220M	64.42	46.58	74.77	58.66	46.04	73.06	28.78	30.55	55.12	58.95	44.72	76.58
JointGT (BART)	160M	65.92	47.15	**76.10****	58.55	45.01	72.31	**30.02***	**32.05****	**55.60**	**65.89****	**48.25****	**78.87****
JointGT (T5)	265M	**66.14****	**47.25****	75.91	**61.01****	**46.32****	**73.57****	28.95	31.29	54.47	60.45	45.38	77.59

Table 1: Results on WebNLG, WebQuestions and PathQuestions. SOTA-NPT indicates the state-of-the-art performance from the baselines without pre-training. #Param means the number of model parameters. The results marked with †, ‡ and ♯ are re-printed from Shimorina and Gardent (2018), Chen et al. (2020d) and Chen et al. (2020b), respectively. - means that the results are not reported in the corresponding references. * indicates that our model significantly outperforms BART and T5 on the corresponding datasets (t-test, $p < 0.05$), while ** means $p < 0.01$.

Wikipedia hyperlinks of entities in the sentences. The detailed statistics of KGTEXT are shown in Table 2.

Dataset	#Ent	#Rel	#Instances (Train / Valid / Test)	#Triples	Length
KGTEXT	1.8M	1,210	6.98M / 10K / 10K	27.2	20.2
WebNLG(U)	3,114	373	34,352 / 4,316 / 4,224	2.9	22.7
WebNLG(C)	3,129	373	34,536 / 4,217 / 4,148	2.9	19.8
WebQuestions	25,703	672	18,989 / 2,000 / 2,000	5.8	15.0
PathQuestions	7,250	378	9,793 / 1,000 / 1,000	2.7	14.0

Table 2: Statistics of pre-training and fine-tuning datasets, including the total number of entities and relations, the data split, the average number of triples, and the average length of texts.

Since our model can adapt to Transformer-based pre-trained models with the encoder-decoder framework, we chose BART (Lewis et al., 2020) and T5 (Raffel et al., 2020) as the base model in this paper, which are denoted by JointGT (BART) and JointGT (T5), respectively. The hyper-parameters of the Transformer blocks were the same as BART-base and T5-base because of the limited computational resources. We initialized our model parameters with the pre-trained checkpoint of BART-base / T5-base except for the structure-aware semantic aggregation module, which was randomly initialized. We followed BART / T5 to use Byte-Pair Encoding (BPE) vocabulary (Radford et al., 2019) with the size of 50,265 / WordPiece vocabulary (Kudo and Richardson, 2018) with the size of 32,000. The batch size was 42 / 32 for JointGT (BART) / JointGT (T5). The maximum length of linearized input graphs was 600, while the maximum length of text sequences was 64. We adopted Adam (Kingma and Ba, 2015) as the optimizer and set the learning rate to be 3e-5. The warmup ratio was 0.1. JointGT was pre-trained on KGTEXT for 1 epoch with the proposed pre-training tasks. It took 44 / 69 hours for JointGT (BART) / JointGT (T5) on 3 NVIDIA Quadro RTX 6000 GPUs.

4.2 Fine-Tuning Settings

We adopted WebNLG, WebQuestions and Path Questions as the benchmark datasets during fine-tuning, and provided the statistics in Table 2.

WebNLG: This dataset aims to convert RDF triples into a textual description. We followed the existing work (Chen et al., 2020b) to use the version of 2.0 (Shimorina and Gardent, 2018). This dataset contains two official data splits: the traditional split (Unconstrained) which guarantees that there is no overlap of input graphs among train / validation / test sets, and a more challenging split (Constrained) where the non-overlap constraint is applied to the triples of input graphs. We denoted these two data splits as *WebNLG(U)* and *WebNLG(C)* in our paper. We followed the preprocessing steps of the existing work (Chen et al., 2020b) to replace the underlines in the entities and relations with spaces, and split the entities and relations in a camel case into multiple words.

WebQuestions: This dataset (Yih et al., 2016; Talmor and Berant, 2018) is the benchmark for question generation over knowledge bases (KBQG), whose purpose is to generate natural language questions about the corresponding knowledge graphs (Serban et al., 2016). It is constructed from two question answering datasets, i.e., WebQuestionsSP (Yih et al., 2016) and ComplexWebQuestions (Talmor and Berant, 2018). These two datasets contain natural language questions, SPARQL queries and answer entities. We converted the SPARQL query to return a subgraph, and used the same preprocessing steps and data splits as the existing work (Kumar et al., 2019; Chen et al., 2020d).

PathQuestions: Similar to WebQuestions, the PathQuestions dataset is also the benchmark for KBQG, which is constructed from a question answering dataset (Zhou et al., 2018b). The main

Model	Fluency			κ	Adequacy			κ
	Win (%)	Lose (%)	Tie (%)		Win (%)	Lose (%)	Tie (%)	
JointGT (BART) vs. BART	29.0*	19.7	51.3	0.413	26.3**	16.0	57.7	0.517
JointGT (T5) vs. T5	23.7	18.7	57.6	0.405	22.7*	16.3	61.0	0.424

Table 3: Human evaluation on WebNLG(U). The scores indicate the percentages of win, lose and tie when JointGT is compared with other baselines. κ is Fleiss' Kappa (all indicate moderate agreement). The scores marked with * mean $p < 0.05$ while ** means $p < 0.01$ in sign test.

difference is that the knowledge graph in PathQuestions is a 2-hop / 3-hop path between two entities. We used the same preprocessing steps and data splits as the existing work (Kumar et al., 2019; Chen et al., 2020d).

More detailed fine-tuning settings including the search space and the best assignment of hyper-parameters on the downstream datasets are reported in the Appendix B.

4.3 Baselines

We chose the following two categories of models as our baselines:

Pre-Trained Models: We adopted KGPT (Chen et al., 2020b), BART (Lewis et al., 2020) and T5 (Raffel et al., 2020) as the pre-trained baselines. KGPT is a pre-trained model for KG-to-text generation, which utilizes the same pre-training dataset as our model and directly uses KG-to-text generation as the pre-training task. BART and T5, as the state-of-the-art pre-trained models for text generation, can be applied to KG-to-text generation with the input of linearized knowledge graphs and the output of text sequences (Ribeiro et al., 2020a).

Task-Specific Models without Pre-Training: We also chose the state-of-the-art task-specific models without pre-training for each dataset as our baselines, including Seq2Seq with copying or delexicalisation (Shimorina and Gardent, 2018) for WebNLG v2.0, and G2S (Chen et al., 2020d) for WebQuestions and PathQuestions.

We directly re-printed the results of baselines if they use the same datasets as ours. Otherwise, we implemented the baselines based on the codes and model parameters released by the original papers. We reported all the results of our implemented models with the mean values over 5 runs.

4.4 Automatic Evaluation

We followed the existing work (Shimorina and Gardent, 2018; Chen et al., 2020d) to use BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005) and ROUGE-L (Lin, 2004) as our automatic metrics. The main results on WebNLG, WebQuestions and PathQuestions are shown in Table 1. We can observe that JointGT based on BART / T5 can outperform vanilla BART / T5 on most of the metrics, respectively, and obtain the state-of-the-art performance on all the datasets. This indicates that our method can promote graph-text alignments and further enhance the performance of the state-of-the-art pre-trained models on KG-to-text datasets.

4.5 Human Evaluation

To further evaluate the quality of generated results, we conducted human evaluation on the WebNLG(U) dataset. We followed the existing work (Ferreira et al., 2019; Ribeiro et al., 2020b) to select two criteria: *fluency* (whether a sentence is grammatically fluent) and *adequacy* (whether a sentence clearly describes the knowledge graph). We randomly sampled 100 knowledge graphs from the test set, and collected the generated results from our models and the most competitive baseline models (i.e., BART and T5). We used the pairwise comparison between BART / T5 and JointGT (BART) / JointGT (T5). Specifically, for each pair of generated texts (one from JointGT and the other from the corresponding baseline, given the same input knowledge graph), three annotators were hired to label which text is better (i.e., win, lose or tie) in terms of the metrics mentioned above. Note that the two metrics were evaluated independently.

Results in Table 3 show that JointGT can beat the corresponding baselines in both fluency and adequacy. Especially for adequacy, our model can significantly outperform BART / T5, which indicates that our model equipped with the structure-aware encoder and well-designed pre-training tasks can generate high-quality texts to describe knowledge graphs more clearly. To evaluate the agreement among different annotators, we calculated Fleiss' Kappa (Fleiss, 1971) for each pairwise comparison, where the results in Table 3 show moderate agreement ($0.4 \leq \kappa \leq 0.6$).

4.6 Ablation Study

4.6.1 Encoder Structure

To investigate the effect of our proposed structure-aware semantic aggregation module, we fixed the pre-training tasks and compared our encoder with two Transformer-based encoders commonly used in the existing work:

SeqEnc: This sequence encoder takes linearized graphs as input and ignores structural information (Ribeiro et al., 2020a; Kale and Rastogi, 2020).

RelEnc: This relation-aware encoder regards the entity sequence as input and leverages the relation embedding into the self-attention layer. Both the entity and relation embedding vectors are directly learned as model parameters (Shaw et al., 2018; Zhu et al., 2019; Song et al., 2020).

Model	#Param	BLEU	METEOR	ROUGE
JointGT (BART)	160M	**65.92**	**47.15**	**76.10**
w/ SeqEnc	140M	64.82	46.87	75.37
w/ RelEnc	160M	65.17	47.07	75.69

Table 4: Ablation test of different encoder structures on WebNLG(U), including our encoder, sequence encoder (SeqEnc) and relation-aware encoder (RelEnc).

Note that we only chose the encoder structures that can directly adapt to BART / T5 for fair comparison[5]. Results in Table 4 show that our encoder structure can perform better than the other baselines. Compared with the relation-aware encoder which can also capture the structural information of knowledge graphs, our model fully utilizes the effective contextual semantic representation to initialize the entity / relation representation at each Transformer layer instead of directly using the learnable entity / relation embedding vectors. This design equips JointGT with better generalization ability during fine-tuning, thereby enhancing our performance on downstream datasets.

Model	#Triples	
	1-3	4-7
JointGT (BART)	**71.24**	**61.36**
w/ SeqEnc	70.83 (-0.41)	60.11 (-1.25)
w/ RelEnc	70.98 (-0.26)	60.58 (-0.78)

Table 5: BLEU scores of three encoders on the test set of WebNLG(U) with different numbers of input triples.

To further demonstrate the effectiveness of our encoder, we divided the test set of WebNLG(U) into two subsets according to the number of triples in knowledge graphs, and compared the performance of three encoders. Results in Table 5 show that the improvement margin between our encoder and other encoders is more evident when the number of input triples is large, which indicates that our model can facilitate the encoding of knowledge graphs with more complex structures.

[5]We observed a significant performance drop if we used the encoders which are incompatible with BART / T5 (such as graph neural networks) because we had to randomly initialize the parameters of them during pre-training.

4.6.2 Pre-Training Task

Model	BLEU	METEOR	ROUGE
JointGT (BART)	**65.92**	**47.15**	**76.10**
w/o TextRecon	64.22	46.56	74.96
w/o GraphRecon	65.37	47.09	75.97
w/o OT	65.03	47.09	75.83
w/ BARTPretrain	64.60	46.78	75.74
w/ KGPTPretrain	65.14	46.94	75.72

Table 6: Ablation test of three pre-training tasks on WebNLG(U), including text / graph reconstruction and graph-text alignments via OT. BARTPretrain / KGPTPretrain means using the pre-training tasks of BART / KGPT instead of our tasks on KGTEXT.

To study the effect of three pre-training tasks, we maintained the encoder structure and removed each task respectively to test the performance. We also replaced all our pre-training tasks with the tasks of the existing work for comparison:

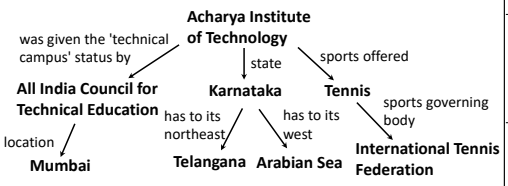
BARTPretrain: The pre-training tasks of BART including text infilling and sentence permutation (Lewis et al., 2020). Since these tasks cannot be applied to graph data, we only used these tasks on the text data of the pre-training dataset.

KGPTPretrain: The pre-training task of KGPT, i.e., KG-to-text generation on the pre-training dataset (Chen et al., 2020b).

Results in Table 6 show that each of our pre-training tasks contributes to the model performance. Compared with the other two tasks, graph enhanced text reconstruction plays a more important role in the task of KG-to-text generation, which directly supervises the decoder with the conditional generation loss. We also observe an apparent performance drop if we replace our pre-training tasks with those proposed by the existing work, thereby indicating the effectiveness of our pre-training tasks to promote KG-to-text generation.

4.7 Few-Shot Learning

To further analyze whether our pre-training tasks can learn a good graph-text joint representation that benefits the downstream KG-to-text generation tasks, we considered the few-shot setting where



BART: The Acharya Institute of Technology is located in the state of Karnataka which has Telangana to its northeast and the Arabian Sea to its west . The Institute was given the 'Technical Campus ' status by the All India Council for Technical Education in Mumbai . The International Tennis Federation governs tennis . **[sports offered]**

T5: The Acharya Institute of Technology is located in Karnataka , India . It was given the 'Technical Campus ' status by the All India Council for Technical Education in Mumbai . The Institute is affiliated with the International Tennis Federation . The Arabian Sea is to the west of Karnataka and Telangana is northeast of the state . Tennis is one of the sports offered by the Institute . **[sports governing body]**

JointGT (BART): The Acharya Institute of Technology is located in the state of Karnataka which has Telangana to its northeast and the Arabian Sea to its west . The Institute was given the 'Technical Campus ' status by the All India Council for Technical Education in Mumbai . One of the sports offered at the Institute is tennis which is governed by the International Tennis Federation .

JointGT (T5): The Acharya Institute of Technology is located in the state of Karnataka . Karnataka has Telangana to its northeast and the Arabian Sea to its west . The Institute was given the 'Technical Campus ' status by the All India Council for Technical Education in Mumbai . The Institute offers tennis which is governed by the International Tennis Federation .

Figure 4: Generated results on WebNLG(U). We highlight the missing and unfaithful parts of each text in red and blue, respectively.

Model	Data Proportion			
	0.5%	1%	5%	10%
BART	33.92	39.08	52.24	56.58
JointGT (BART)	**37.18**	**42.26**	**54.41**	**57.73**
w/ BARTPretrain	32.63	37.11	52.91	56.81
w/ KGPTPretrain	35.33	40.72	53.08	57.18

Table 7: BLEU scores of the models with corresponding pre-training tasks trained on different proportions of WebNLG(U).

only a few training instances were used during fine-tuning. We still fixed our model structure and compared our pre-training tasks with the tasks of BART and KGPT mentioned in §4.6.2.

Results in Table 7 show that our pre-training tasks can perform better than other tasks, especially when the amount of training data is small. This indicates that our proposed tasks can capture the graph-text alignments during pre-training, thereby making our model generalizable to the downstream KG-to-text datasets better with only a few training samples.

4.8 Case Study

To intuitively show the generation quality of our model, we provided some generated cases in Figure 4. We observe that JointGT can generate high-quality texts that describe the knowledge graph more completely and faithfully. For example, in the generated case on WebNLG(U), both BART and T5 fail to cover all the input triples, where BART misses the triple *(Acharya Institute of Technology, sports offer, Tennis)* and T5 misses *(Tennis, sports governing body, International Tennis Federation)*. Also, T5 generates non-existing facts that are unfaithful to the knowledge graph. Equipped with the structure-aware Transformer encoder and the well-designed pre-training tasks to learn graph-text alignments, JointGT (BART) and JointGT (T5) can generate descriptions which include all the input triples and express the relation between each pair of entities more faithfully.

5 Conclusion

We propose a novel graph-text joint representation learning model called JointGT for KG-to-text generation. This model plugs a simple structure-aware semantic aggregation module into the vanilla Transformer layer to preserve the structure of input graphs, and utilizes three pre-training tasks to learn graph-text alignments in the discrete vocabulary space and continuous embedding space. Experiments show that JointGT can outperform state-of-the-art pre-trained NLG models on various datasets of KG-to-text generation.

Acknowledgments

This work was partly supported by the NSFC projects (Key project with No. 61936010 and regular project with No. 61876096). This work was also supported by the Guoqiang Institute of Tsinghua University, with Grant No. 2019GQG1 and 2020GQG0005.

References

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: an automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization@ACL 2005*, pages 65–72.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko.

2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795.

Liqun Chen, Zhe Gan, Yu Cheng, Linjie Li, Lawrence Carin, and Jingjing Liu. 2020a. Graph optimal transport for cross-domain alignment. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 1542–1553.

Wenhu Chen, Yu Su, Xifeng Yan, and William Yang Wang. 2020b. KGPT: knowledge-grounded pre-training for data-to-text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 8635–8648.

Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. 2020c. UNITER: universal image-text representation learning. In *ECCV*, volume 12375, pages 104–120.

Yu Chen, Lingfei Wu, and Mohammed J. Zaki. 2020d. Toward subgraph guided knowledge graph question generation with graph neural networks. *arXiv preprint arXiv: 2004.06015*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186.

Thiago Castro Ferreira, Chris van der Lee, Emiel van Miltenburg, and Emiel Krahmer. 2019. Neural data-to-text generation: A comparison between pipeline and end-to-end architectures. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 552–562.

J. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76:378–382.

Zihao Fu, Bei Shi, Wai Lam, Lidong Bing, and Zhiyuan Liu. 2020. Partially-aligned data-to-text generation with distant supervision. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 9183–9193.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The webnlg challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133.

Jian Guan, Yansen Wang, and Minlie Huang. 2019. Story ending generation with incremental encoding and commonsense knowledge. In *The Thirty-Third AAAI Conference on Artificial Intelligence*, pages 6473–6480.

Qipeng Guo, Zhijing Jin, Xipeng Qiu, Weinan Zhang, David Wipf, and Zheng Zhang. 2020. Cyclegt: Unsupervised graph-to-text and text-to-graph generation via cycle training. *arXiv preprint arXiv: 2006.04702*.

Haozhe Ji, Pei Ke, Shaohan Huang, Furu Wei, Xiaoyan Zhu, and Minlie Huang. 2020. Language generation with multi-hop reasoning on commonsense knowledge graph. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 725–736.

Zhijing Jin, Qipeng Guo, Xipeng Qiu, and Zheng Zhang. 2020. Genwiki: A dataset of 1.3 million content-sharing text and graphs for unsupervised graph-to-text generation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2398–2409.

Mihir Kale and Abhinav Rastogi. 2020. Text-to-text pre-training for data-to-text tasks. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 97–102.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text generation from knowledge graphs with graph transformers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2284–2293.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018: System Demonstrations*, pages 66–71.

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 *18th International Semantic Web Conference*, volume 11778, pages 382–398.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Manuel Mager, Ramón Fernandez Astudillo, Tahira Naseem, Md. Arafat Sultan, Young-Suk Lee, Radu Florian, and Salim Roukos. 2020. Gpt-too: A language-model-first approach for amr-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1846–1852.

Diego Marcheggiani and Laura Perez-Beltrachini. 2018. Deep graph convolutional encoders for structured data to text generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 1–9.

Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019. Step-by-step: Separating planning from realization in neural data-to-text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2267–2277.

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 of the Association for Computational Linguistics*, pages 311–318.

Matthew E. Peters, Mark Neumann, Robert L. Logan IV, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge enhanced contextual word representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 43–54.

Gabriel Peyré and Marco Cuturi. 2019. Computational optimal transport. *Found. Trends Mach. Learn.*, 11(5-6):355–607.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. In *OpenAI Technical Report*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. In *OpenAI Technical Report*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Leonardo F. R. Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. 2020a. Investigating pretrained language models for graph-to-text generation. *arXiv preprint arXiv: 2007.08426*.

Leonardo F. R. Ribeiro, Yue Zhang, Claire Gardent, and Iryna Gurevych. 2020b. Modeling global and local node contexts for text generation from knowledge graphs. *Trans. Assoc. Comput. Linguistics*, 8:589–604.

Martin Schmitt, Leonardo F. R. Ribeiro, Philipp Dufter, Iryna Gurevych, and Hinrich Schütze. 2020a. Modeling graph structure via relative position for better text generation from knowledge graphs. *arXiv preprint arXiv: 2006.09242*.

Martin Schmitt, Sahand Sharifzadeh, Volker Tresp, and Hinrich Schütze. 2020b. An unsupervised joint system for text generation from knowledge graphs and semantic parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 7117–7130.

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

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 464–468.

Anastasia Shimorina and Claire Gardent. 2018. Handling rare items in data-to-text generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 360–370.

Linfeng Song, Ante Wang, Jinsong Su, Yue Zhang, Kun Xu, Yubin Ge, and Dong Yu. 2020. Structural information preserving for graph-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7987–7998.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 641–651.

Bayu Distiawan Trisedya, 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*, pages 1627–1637.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. KEPLER: A unified model for knowledge embedding and pre-trained language representation. *Trans. Assoc. Comput. Linguistics*, 9:176–194.

Yujia Xie, Xiangfeng Wang, Ruijia Wang, and Hongyuan Zha. 2019. A fast proximal point method for computing exact wasserstein distance. In *Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence*, volume 115, pages 433–453.

Wen-tau Yih, Matthew Richardson, Christopher 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*.

Donghan Yu, Chenguang Zhu, Yiming Yang, and Michael Zeng. 2020. JAKET: joint pre-training of knowledge graph and language understanding. *arXiv preprint arXiv: 2010.00796*.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: enhanced language representation with informative entities. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 1441–1451.

Chao Zhao, Marilyn A. Walker, and Snigdha Chaturvedi. 2020. Bridging the structural gap between encoding and decoding for data-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2481–2491.

Hao Zhou, Tom Young, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2018a. Commonsense knowledge aware conversation generation with graph attention. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pages 4623–4629.

Mantong Zhou, Minlie Huang, and Xiaoyan Zhu. 2018b. An interpretable reasoning network for multi-relation question answering. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2010–2022.

Jie Zhu, Junhui Li, Muhua Zhu, Longhua Qian, Min Zhang, and Guodong Zhou. 2019. Modeling graph structure in transformer for better amr-to-text generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5458–5467.

A IPOT Algorithm

Inexact Proximal point method for Optimal Transport (IPOT) is an effective iterative method to approximate OT distance and compute the transport plan $\boldsymbol{T}$ (Xie et al., 2019). Given the sequence of entities and relations in the knowledge graph $\mathcal{G}_{seq} = (g_1, \cdots, g_{|\mathcal{V}|+|\mathcal{E}|})$ with its corresponding embedding vectors $\boldsymbol{H}^{\mathcal{G}} = (\boldsymbol{h}_1^{\mathcal{G}}, \cdots, \boldsymbol{h}_{|\mathcal{V}|+|\mathcal{E}|}^{\mathcal{G}})$, and the text sequence $X = (x_1, \cdots, x_n)$ with its embedding vectors $\boldsymbol{S} = (\boldsymbol{s}_1, \cdots, \boldsymbol{s}_n)$, the implementation of IPOT algorithm to calculate $\boldsymbol{T}$ is shown in Algorithm 1.

Algorithm 1 IPOT Algorithm

Require:
$\mathcal{G}_{seq} = \{g_i\}_{i=1}^{|\mathcal{V}|+|\mathcal{E}|}$, $X = \{x_j\}_{j=1}^{n}$, and their embedding vectors $\boldsymbol{H}^{\mathcal{G}} = \{\boldsymbol{h}_i^{\mathcal{G}}\}_{i=1}^{|\mathcal{V}|+|\mathcal{E}|}$, $\boldsymbol{S} = \{\boldsymbol{s}_j\}_{j=1}^{n}$
Generalized stepsize: $1/\beta$

$\boldsymbol{\sigma} = \frac{1}{n}\mathbf{1}_n$, $\boldsymbol{T}^{(1)} = \mathbf{1}_{|\mathcal{V}|+|\mathcal{E}|}\mathbf{1}_n^\top$
$\boldsymbol{C}_{ij} = d(g_i, x_j) = 1 - \frac{\boldsymbol{h}_i^{\mathcal{G}}\boldsymbol{s}_j}{\|\boldsymbol{h}_i^{\mathcal{G}}\|_2\|\boldsymbol{s}_j\|_2}$
$\boldsymbol{A}_{ij} = e^{-\frac{\boldsymbol{C}_{ij}}{\beta}}$
for $t = 1$ to N **do**
  $\boldsymbol{Q} = \boldsymbol{A} \odot \boldsymbol{T}^{(t)}$
  for $k = 1$ to K **do**
    $\boldsymbol{\delta} = \frac{1}{(|\mathcal{V}|+|\mathcal{E}|)\boldsymbol{Q}\boldsymbol{\sigma}}, \boldsymbol{\sigma} = \frac{1}{n\boldsymbol{Q}^\top\boldsymbol{\delta}}$
  end for
  $\boldsymbol{T}^{(t+1)} = \mathrm{diag}(\boldsymbol{\delta})\boldsymbol{Q}\mathrm{diag}(\boldsymbol{\sigma})$
end for
return $\boldsymbol{T}$

In the algorithm of IPOT, $\odot$ denotes Hadamard product. β, K and N are all hyper-parameters. We followed the existing work (Chen et al., 2020a) to set $\beta = 1.0$, $K = 1$ and $N = 10$.

B Hyper-Parameter Setting

Hyper-parameter	Search Space
Masking Probability (entity / relation / word)	*choice*[20%,30%,40%]
Learning Rate	*choice*[2e-5,3e-5,5e-5]
Training Epoch	*choice*[1,2]
Warmup Ratio	*choice*[0,0.1]
Batch Size	*choice*[32,36,42]
Input Length	600
Output Length	64
Maximum Gradient Norm	1.0
Optimizer	Adam
Epsilon (for Adam)	1e-8

Table 8: Hyper-parameter search space of JointGT during pre-training. *choice* indicates that the listed numbers will be chosen with the same probability.

We provided the detailed settings of hyper-parameters during pre-training and fine-tuning. The settings include hyper-parameter search space and best assignments. Note that we used Huggingface's Transformers[6] to implement our models.

[6] https://github.com/huggingface/transformers

Hyper-parameter	Search Space
Learning Rate	*choice*[2e-5,3e-5,5e-5,1e-4]
Training Epoch	*choice*[20,30,40]
Warmup Step	*uniform-integer*[0,total_step*0.2]
Batch Size	*choice*[24,32]
Input Length	*choice*[128,256]
Output Length	*choice*[64,128]
Beam Size	*choice*[2,3,5]
Length Penalty	*choice*[1.0,3.0,5.0]
Maximum Gradient Norm	1.0
Optimizer	Adam
Epsilon (for Adam)	1e-8

Table 9: Hyper-parameter search space of JointGT during fine-tuning. *uniform-integer* means the integers in the interval can be selected uniformly. In the search space of warmup step, total_step denotes the total training steps on the corresponding datasets.

Thus all the hyper-parameters reported in our paper were consistent with the codes of Huggingface's Transformers.

Model	JointGT (BART)			
Dataset	WebNLG(U)	WebNLG(C)	WebQuestions	PathQuestions
Learning Rate	2e-5	2e-5	2e-5	5e-5
Training Epoch	40	20	30	40
Warmup Step	1,600	0	3,400	1,100
Batch Size	32	32	32	32
Input Length	256	256	256	128
Output Length	128	128	128	64
Beam Size	5	5	5	5
Length Penalty	1.0	1.0	5.0	1.0
Model	JointGT (T5)			
Dataset	WebNLG(U)	WebNLG(C)	WebQuestions	PathQuestions
Learning Rate	5e-5	3e-5	1e-4	2e-5
Training Epoch	30	30	40	30
Warmup Step	1,600	1,200	2,300	900
Batch Size	24	32	32	32
Input Length	256	256	256	128
Output Length	128	128	64	64
Beam Size	5	5	5	2
Length Penalty	1.0	1.0	5.0	1.0

Table 10: Best assignments of hyper-parameters on the downstream datasets.

We presented the hyper-parameter search space during pre-training in Table 8. The number of hyper-parameter search trials was 10. Manual search was adopted to select hyper-parameters, and the selection criterion was BLEU on the validation set when we fine-tuned the pre-trained model on WebNLG(U). The best assignment of pre-training was described in our main content.

We also provided the detailed settings of hyper-parameters during fine-tuning on the downstream datasets, including the hyper-parameter search space in Table 9 and the best assignments in Table 10. The number of hyper-parameter search trials was 20. BLEU was adopted as our criterion in the manual search on all the downstream tasks.