Exploring the Synergy of Dual-path Encoder and Alignment Module for Better Graph-to-Text Generation

Tianxin Zhao[1,2,3,*], **Yingxin Liu**[1,2,3,*], **Xiangdong Su**[1,2,3(✉)], **Jiang Li**[1,2,3], **Guanglai Gao**[1,2,3]
[1]College of Computer Science, Inner Mongolia University, Hohhot, China
[2]National & Local Joint Engineering Research Center of
Intelligent Information Processing Technology for Mongolian
[3]Inner Mongolia Key Laboratory of Mongolian Information Processing Technology
zhaotianx@foxmail.com,sianliu@mail.imu.edu.cn, cssxd@imu.edu.cn

Abstract

The mainstream approaches view the knowledge graph-to-text (KG-to-text) generation as a sequence-to-sequence task and fine-tune the pre-trained model (PLM) to generate the target text from the linearized knowledge graph. However, the linearization of knowledge graphs and the structure of PLMs lead to the loss of a large amount of graph structure information. Moreover, PLMs lack an explicit graph-text alignment strategy because of the discrepancy between structural and textual information. To solve these two problems, we propose a synergetic KG-to-text model with a dual-path encoder, an alignment module, and a guidance module. The dual-path encoder consists of a graph structure encoder and a text encoder, which can better encode the structure and text information of the knowledge graph. The alignment module contains a two-layer Transformer block and an MLP block, which aligns and integrates the information from the dual encoder. The guidance module combines an improved pointer network and an MLP block to avoid error-generated entities and ensures the fluency and accuracy of the generated text. Our approach obtains very competitive performance on three benchmark datasets. Our code is available from https://github.com/IMU-MachineLearningSXD/G2T.

Keywords: KG-to-text Generation, Dual-path Encoder, Feature Alignment, Pointer Network

1. Introduction

Knowledge graph (KG) is a graph-structured knowledge base to store real-world entities and the relationships between them. The KG-to-text generation task (Gardent et al., 2017) aims to generate high-quality texts that are consistent with input knowledge graphs, in which the generation model needs to encode the structure and content of the knowledge graphs and decode the latent representation into the target text effectively. In practical application scenarios, KG-to-text can further enhance the capability of natural language processing systems to understand better and utilize knowledge graphs. Figure 1 shows an example of KG-to-text generation, which includes a knowledge graph, the linearized sequence of the specific knowledge graph, and the target natural language description of such knowledge graph.

Due to the sparseness of the available KG-to-text datasets, it is challenging for typical text generation models to learn the alignment relationship between knowledge graphs and target tokens from scratch. Concerning the fact that pre-trained language models (PLMs) have learned rich linguistic knowledge, contextual information, and semantic relationships from the pre-training corpus, current approaches generally transform KG-to-text generation into a sequence-to-sequence task and explore the use of PLMs as the generation models for this tasks. They fine-tune the PLMs on the training dataset, in which the knowledge graph is linearized into an input sequence of PLMs.

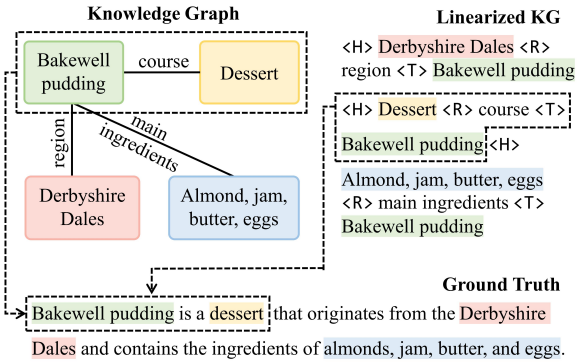

Figure 1: The knowledge graph (subgraph), linearized knowledge graph (subsequence), and corresponding text description (clause). The related parts of the knowledge graph and sequence have been marked with the same color. The special tokens <H>, <R>, and <T> mean the head entity, relation, and tail entity in the knowledge triples, respectively.

*First Author and Second Author contribute equally to this work.

LREC-COLING 2024, pages 6980–6991

Although fine-tuning PLMs has yielded promising results, there are still several issues. Firstly, existing methods lose a significant amount of graph structure information, since they use linearized knowledge graphs as input. Although some efforts have modified the encoders of PLMs to incorporate graph structure information, these methods still rely on sequence-to-sequence models and cannot effectively utilize the complete graph structure. Secondly, PLMs lack explicit graph-text alignment. Current PLMs typically employ auto-encoder or auto-regressive methods to learn text alignment relationships. In KG-to-text generation, the alignment between knowledge graphs and texts is often not a simple one-to-one correspondence but rather one-to-many or even many-to-many correspondence, making it significantly challenging to establish accurate alignment between knowledge graphs and texts.

To address the two issues in KG-to-text generation, we explore the synergy of the dual-path encoder and an alignment module in the proposed model. For the problem of losing graph structure information, we design a dual-path encoder, which comprises a graph encoder based on graph attention (GAT) and a text encoder based on PLMs. The dual-path encoder simultaneously takes the knowledge graph and the linearized knowledge graph as input, encoding both the structural and textual information of the knowledge graphs. For the problem of lacking graph and text alignment, we designed an alignment module after the dual-path encoder. This module, based on Transformer, aligns the text and graph information and merges them. Additionally, we incorporate a guidance module to ensure the quality of the generated text, which involves an improved pointer network and an MLP block. We conducted experiments on three publicly available KG-to-text datasets, and the results demonstrate that our model is highly competitive among current KG-to-text models.

In summary, our innovation lies in proposing a synergistic KG-to-text model composed of three main components:

- **Dual-path Encoder** In our proposed model, the dual-path encoder consists of a graph encoder based on graph self-attention (GAT), and a text encoder based on a pre-trained model (PLM), which can better capture the structure and text information of the knowledge graph and avoid the problem of structure information loss.
- **Alignment Module** The alignment module based on Transformer allows the text information and graph information to be better aligned and fused.
- **Guidance Module** The guidance module employs an improved pointer network to avoid error-generated entities and ensures the fluency and accuracy of the generated text.

2. Related Work

2.1. KG-to-Text Generation

Since the WebNLG dataset was proposed (Gardent et al., 2017), KG-to-text generation has become a popular task. Early tasks proposed a neural network-based approach to generate text from linearized KG triples (Gardent et al., 2017), but the approach could not model the knowledge graph's structural information. To obtain information about the structure of knowledge graphs. (1) Some work focuses on the use of graph neural networks (GNNs) or graph transformers that explicitly encode graph structure (Velickovic et al., 2017; Koncel-Kedziorski et al., 2019). (2) Some work uses non-parallel graphical data and designs unsupervised training targets as a way to jointly learn graph-to-text and text-to-graph conversion tasks (Schmitt et al., 2020; Guo et al., 2020; Jin et al., 2020). (3) Some work has explored the linearization order of knowledge graphs. Most of the work is based on heuristic graph traversal methods for KG sequential generation (Flanigan et al., 2016; Gardent et al., 2017; Ke et al., 2021). (Li et al., 2021) proposed to linearize the KG using a relational biased breadth-first search (RBFS) strategy. However, none of these works consider lexical order information in the ground truth. (Liu et al., 2022) extracts order information from the ground truth as supervision and trains an order prediction module to generate the optimal order that captures the partial graph structure in the triples.

2.2. KG-to-Text Generation Based on PLMs

In recent years, PLMs have made remarkable achievements in natural language processing tasks (Radford et al., 2018; Devlin et al., 2019; Lewis et al., 2020; Raffel et al., 2020). After pre-training with a large-scale corpus, pre-trained models demonstrate unprecedented generalization capabilities to solve relevant downstream tasks. These models have been adapted and fine-tuned for the KG-to-text task. (Zhang et al., 2019; Peters et al., 2019) use fixed entity embeddings based on TransE (Bordes et al., 2013) or word vectors (Mikolov et al., 2013)directly during the pre-training process. (Ribeiro et al., 2020) fine-tune PLMs to leverage the generative power of PLMs. KEPLER (Wang et al., 2021) and JAKET (Yu et al., 2022), on the other hand, use a joint pre-trained graph-to-text representation approach. Specifically, they encode textual information of entities

using pre-trained language models and jointly optimize knowledge embedding targets and mask language modeling targets. Unlike the joint graph-text coding approaches considered for natural language understanding tasks, (Ke et al., 2021; Colas et al., 2022) focus on a joint pre-training approach for knowledge graph coding and sequence decoding in KG-to-Text generation tasks. Different from them, we add a graph encoder in addition to the text encoder. The graph encoder captures structural information directly using the knowledge graph as input. Our model uses a dual-path encoder.

3. Method

3.1. Overview Of The Proposed Model

KG-to-text generation is the process of converting structured knowledge from the knowledge graph into natural language. We are given the knowledge graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{e_1, e_2, \ldots, e_{|\mathcal{V}|}\}$ is the set of entities containing the knowledge graph, $\mathcal{E} = (r_{ij})_{|\mathcal{V}|\times|\mathcal{V}|}$ is a set containing the relationships of entities in the graph. The linearized knowledge graph $\mathcal{G}_{\text{linear}} = (l_1, l_2, \ldots, l_m)$ is a sequence of all the triples in the knowledge graph into a sequence with m tokens. The sequence contains the tokens of the components of the triples (heads, relations, tails). Generate a text sequence $S = (w_1, w_2, ..., w_s)$, where there are s words.

This paper proposes a synergetic KG-to-text model with a dual-path encoder, an alignment module, and a guidance module. The dual-path encoder consists of a graph structure encoder and a text encoder, which can better encode the structure and text information of the knowledge graph. The alignment module contains a two-layer Transformer block and an MLP block, which aligns and integrates the information from the dual encoder. The guidance module combines an improved pointer network and an MLP block to avoid error-generated entities and assists the decoder in generating fluent and accurate natural language.

3.2. Dual-path Encoder

3.2.1. Graph Encoder

Different from (Chen et al., 2020; Ke et al., 2021; Colas et al., 2022) which only use PLM encoders for linearized knowledge graphs, we use both the graph encoder and the text encoder for knowledge graphs. The knowledge graph is used as the input of the graph encoder. At first, the knowledge graph is passed into the embedding layer to obtain the corresponding embedding representation, and the process is formulated as follows:

$$\begin{aligned} \boldsymbol{h}_{\mathcal{G}} &= \text{EMB}_{\text{graph}}(\mathcal{G}) \\ &= \{\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_{|\mathcal{V}|}\}, \end{aligned} \tag{1}$$

where $\boldsymbol{h} \in \mathbb{R}^{|\mathcal{V}|}$, and $|\mathcal{V}|$ is the number of entities in the knowledge graph. $\vec{h}_i \in \mathbb{R}^n$ and is the feature vector of each node. n is the dimension of the feature vector of each node in the knowledge graph.

Then, we pass the embedding encoding to the graph encoder to obtain the structure representation of the knowledge graph. The graph encoder has two parts: (1) the GAT which is used to obtain the structural information of the knowledge graph, and (2) the pooling layer which is used to aggregate the features of the representation. The whole process can be formulated as follows:

$$\begin{aligned} \boldsymbol{h}_{\text{graph}} &= \text{pooling}(\text{GAT}(\boldsymbol{h}_{\mathcal{G}})) \\ &= \{\vec{h}'_1, \vec{h}'_2, \ldots, \vec{h}'_{|\mathcal{V}|}\}, \end{aligned} \tag{2}$$

where $\vec{h}'_i$ is the feature vector of node i obtained by weighting all neighboring nodes j of node i, and the calculation process of Equation 3 can be expressed in detail as follows:

$$\vec{h}'_i = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i}\alpha_{ij}^k \boldsymbol{W}^k \vec{h}_j\right), \tag{3}$$

where K is the number of multi-headed attention, $\alpha_{ij} = \text{softmax}(\text{Attention}(\boldsymbol{W}\vec{h}_i, \boldsymbol{W}\vec{h}_j)$ is the importance of all neighbor nodes j of node i to node i, and $\boldsymbol{W}$ is the trainable parameter, and σ denotes the activation function.

3.2.2. Text Encoder

Inspired by JointGT (Ke et al., 2021), we use the BART encoder as the encoder of knowledge triple sequences. First, we linearize all the triples of the knowledge graph into a sequence $\mathcal{G}_{\text{linear}}$. This sequence is treated as normal text and passed through the embedding layer to obtain the corresponding embedding encoding, and the process is formulated as follows:

$$\begin{aligned} \boldsymbol{h}_{\mathcal{G}_{\text{linear}}} &= \text{EMB}_{\text{text}}(\mathcal{G}_{\text{linear}}) \\ &= \{\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_m\}, \end{aligned} \tag{4}$$

where $\boldsymbol{h}_{\mathcal{G}_{\text{linear}}} \in \mathbb{R}^m$, m is the number of tokens in $\mathcal{G}_{\text{linear}}$. $\vec{h}_i \in \mathbb{R}^n$, $\vec{h}_i$ is the feature vector of the ith token, and n is the dimension of the feature of each token in the linearized knowledge graph.

After that, we feed the embedding encoding into the text encoder to obtain the sequence representation of the linearized knowledge graph, which is formulated as follows:

$$\begin{aligned} \boldsymbol{h}_{\text{text}} &= \text{Encoder}_{\text{text}}(\boldsymbol{h}_{\mathcal{G}_{\text{linear}}}) \\ &= \{\vec{h}'_1, \vec{h}'_2, \ldots, \vec{h}'_m\}, \end{aligned} \tag{5}$$

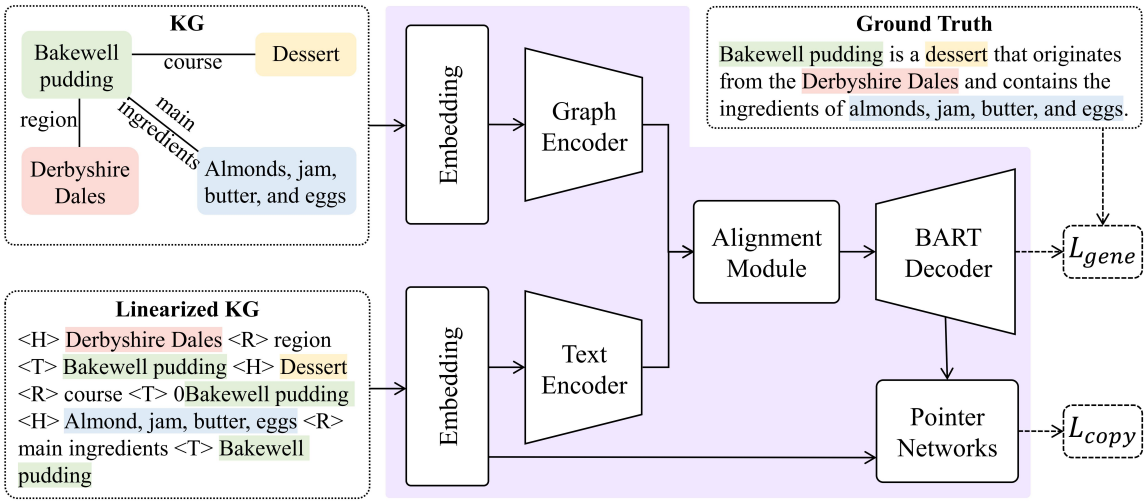

Figure 2: Illustration of the proposed model.

where $\vec{h}'_i \in \mathbb{R}^n$,$\vec{h}'_i$ is the text encoder output of the ith token, and n is the dimension of the hidden vector.

3.3. Alignment Module

After the dual-path encoder, we obtain the graph structure representation and the sequence representation. Then, we concatenate these two representations together and feed them to the alignment module, as

$$\boldsymbol{c} = [\boldsymbol{h}_{\text{text}} \| \lambda \boldsymbol{h}_{\text{graph}}], \tag{6}$$

where h_{text} and h_{graph} are the knowledge graph representations from the graph encoder and text encoder, respectively. λ is the hyperparameter that controls the weight of the graph structure information. Next, the concatenated representation $\boldsymbol{c}$ is processed by a two-layer Transformer block. The computation process is formulated as follows:

$$Q, K, V = \boldsymbol{c}\boldsymbol{W}_Q, \boldsymbol{c}\boldsymbol{W}_K, \boldsymbol{c}\boldsymbol{W}_V, \tag{7}$$

where $\boldsymbol{W}_Q$, $\boldsymbol{W}_K$ and $\boldsymbol{W}_V$ are weight matrices used to compute the query, key, and value for the concatenated representation, respectively.

$$\text{Attention}(\boldsymbol{c}) = \text{Softmax}\left(\frac{Q(K)^T}{\sqrt{d_k}}\right)V, \tag{8}$$

$$\begin{aligned} O_i &= \text{FN}(\text{Attention}(\boldsymbol{c})) \\ &= \text{ReLU}(\text{Attention}(\boldsymbol{c})\boldsymbol{W}_1 + \boldsymbol{b}_1)\boldsymbol{W}_2 + \boldsymbol{b}_2, \end{aligned} \tag{9}$$

where i is the ith attention head, and $\boldsymbol{W}_1$, $\boldsymbol{W}_2$, $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ are the weights and biases of the feed-forward block FN in the Transformer.

After the Transformer block, we further adjust the dimensions and refine the results by passing them through an MLP block composed of fully connected layers. The process is formulated as follows:

$$\boldsymbol{h}_{\text{align}} = \text{FC}(\|_{i=1}^{K} O_i)\boldsymbol{W}_O. \tag{10}$$

where FC denotes the MLP block.

3.4. Guidance Module

To avoid error-generated entities and obtain fluent and accurate results, inspired by (Koncel-Kedziorski et al., 2019; Li et al., 2021), we added a guidance module in the proposed model. The guidance module contains an improved pointer network (See et al., 2017) and an MLP block composed of fully connected layers. We use the pointer network to guide the decoder generation, which requires calculating the probability of copying words from the PLMs generated vocabulary and the knowledge graph text:

$$\text{p}^i_{\text{copy}} = \sigma\left(\boldsymbol{W}\left(\vec{h}_i + \text{EMB}_{text}\left(l_i\right)\right) + \boldsymbol{b}_{\text{copy}}\right), \tag{11}$$

where $\vec{h}_i$ is the ith hidden state in the top layer of the pre-trained model decoder, l_i is the ith token in the linearized knowledge graph, $\boldsymbol{W}$ is the trainable parameter, and $\boldsymbol{b}_{copy}$ is the trainable bias.

Through experiments, we found that adding $\vec{h}_i$ and l_i together can yield better results than the original pointer network that directly collocates $\vec{h}_i$ and l_i. Therefore, we modify the vector calculation to summation when calculating the copying vocabulary probabilities. In addition, we also found in our experiments that using the text encoder embedding encoding is better than using the dual-path encoder hidden state. Therefore, we choose the text encoder embedding encoding as part of the input for the pointer network.

To ensure the word accuracy of the generated text, it is necessary to ensure that the generated vocabulary is derived from the knowledge graph as much as possible. This process is equivalent to minimizing the copy probability p^i_{copy} of the generated tokens w_i from the vocabulary list while maximizing the copy probability p^j_{copy} of the token e_j copied from the knowledge graph entities. Thus,

the loss of the point network is

$$\mathcal{L}_{copy} = \sum_{w_i} \mathrm{p}^{i}_{\mathrm{copy}} + \sum_{e_j} \left(1 - \mathrm{p}^{j}_{\mathrm{copy}}\right), \quad (12)$$

3.5. Loss Function

The decoder of our model is the BART decoder, which is a pre-trained language model. During the model training phase, the primary goal is to minimize the difference between the ground truth and the text generated by the decoder. This process is equivalent to minimizing the negative log-likelihood, as follows:

$$\mathcal{L}_{gene} = -\sum_{i=1}^{T} \log \mathrm{p}_{\mathrm{gene}}\left(w_i \mid w_1, \ldots, w_{i-1}; \mathcal{G}\right), \quad (13)$$

where $\mathrm{p}_{\mathrm{gene}}$ is the generation probability of the pre-trained model.

In the proposed model, the BART decoder aims to generate target text, while the guidance module guarantees the lexical accuracy of the output. Therefore, the total loss function includes the text generation loss and the pointer network loss:

$$\mathcal{L}_{total} = \mathcal{L}_{gene} + \alpha_1 \mathcal{L}_{copy}. \quad (14)$$

where α_1 is a hyper-parameter that is used to trade off these two loss functions.

4. Experiment

4.1. Dataset

In this paper, we used PathQuestions (Zhou et al., 2018) and WebNLG (Gardent et al., 2017) as benchmark datasets to evaluate the performance of our model. The size of the dataset is shown in Table 1.

Dataset	Entities	Relations	Train Size	Test Size
PathQuestions	7250	378	9793	1000
WEBNLG(U)	3114	373	34352	4224
WEBNLG(C)	3129	373	34536	4148

Table 1: The details of the datasets

PathQuestions is constructed using two subsets of Freebase as a knowledge base for the multi-relational question-and-answer task. Each question contains two or three triples per question set and their corresponding textual description. Our work uses the same preprocessing steps and data partitioning approach as in existing work.

WEBNLG is a collection of triples describing facts (entities and relations between them) extracted from DBpedia and the corresponding natural language text, covering about 450 different DBpedia attributes. Each set of triples contains a subject, a point, and an object describing a fact (entity and the relationship between them). Each question in this corpus contains up to 7 triples and one or more corresponding reference text descriptions. It is the most widely used dataset for knowledge graph generation text tasks. We partition the dataset into WebNLG(U) and WebNLG(C) using the same processing steps and the same data partitioning as (Ke et al., 2021).

4.2. Baseline

We used five models of KG-to-Text generation as baselines.

- **BART-Base** (Lewis et al., 2020) linearizes the knowledge graph into a sequence of triples and uses BART-Base to generate the text.
- **T5-Base** (Raffel et al., 2020) linearizes the knowledge graph into a ternary sequence and uses T5-Base to generate the text.
- **KGPT** (Chen et al., 2020) is a pre-trained model based on the knowledge graph for data-to-text generation. This model can be fine-tuned on various data-to-text generation tasks to generate task-specific text.
- **JointGT (BART)** (Ke et al., 2021) is a pre-training model based on joint representation learning of knowledge graphs and text sequences for the KG-to-text task on the KG-TEXT dataset. It uses BART as the base model and adds a structure-aware language aggregation module.
- **GAP** (Colas et al., 2022) is also a pre-trained model for KG-to-text tasks based on joint representation learning of knowledge graphs and text sequences. It is the most advanced model on the WebNLG(U) dataset. It also improves on the encoder of the pre-trained model

4.3. Evaluation Metric

Following (Gardent et al., 2017; Colas et al., 2022), we adopt BLEU(Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), and ROUGE-L (Lin, 2004) as the evaluation metrics. Since BLEU is more concerned with the accuracy and smoothness of the generated results, the BLEU score is also of more significant concern to us.

4.4. Implementation Detail

In the graph encoder, we set the number of layers to 1 and the hidden layer dimension to 768. In the text encoder, we use the improved BART encoder

proposed by JointGT(Ke et al., 2021), which is constructed by adding a graph structure aggregation module to the BART encoder. In the alignment module, we use a two-layer Transformer block with multi-head attention. The maximum length of the linearized input graph is 256, and the maximum length of the output text sequence is 128. We use Adam (Kingma and Ba, 2015) as the optimizer and set the learning rate to $2e-5$. The preheating ratio is 0.1. All parameters are optimized under the supervision of $\mathcal{L}_{total}$. More details can be found in Table 2.

Hyperparameter	PathQuestions	WEBNLG(C)	WEBNLG(U)
Learning Rate	2e-5	2e-5	2e-5
Warmup Steps	300	0	1600
Beam Size	5	5	5
Length Penalty	1.0	1.0	1.0
Num Nodes	50	50	50
Num Relations	60	60	60
Embedding	768	768	768
λ	0.1	0.1	0.1
α_1	0.6	0.6	0.43
Batch Size	32	48	48

Table 2: Hyperparameter details

5. Result and Discussion

5.1. Main Results

To evaluate the effectiveness of the proposed approach, we compare it with the baselines on the above-mentioned three benchmark datasets, including PathQuestions, WebNLG(C), and WebNLG(U). The results are shown in Table 3. On these three datasets, our model outperforms the baselines overall. On the PathQuestions dataset, our model improves the BLEU score by 3.46% and 8.25% over the current state-of-the-art pre-trained models Bart and T5, respectively, and enhances the ROUGE-L score by 1.86% and 3.04%, respectively. Compared with JointGT, our model improves the BLEU and Rouge-L scores by 1.31% and 0.75%, respectively. This indicates the superiority of the proposed model in this paper. According to our analysis, this is because our proposed dual-path encoder not only inherits the capability of the pre-trained model in text encoding but also encodes the graph structure information effectively. Meanwhile, the alignment module can align and fuse the graph representation and text representation from the dual-path encoder effectively. In addition, the proposed guidance module can avoid error-generated entities and ensure the fluency and accuracy of the generated text. On the other two datasets, we achieved the same significant results.

While KGPT and JointGT rely on additional pre-training tasks to encode the graph structure information, our approach adapts the graph structure by a dual-path encoder and requires no additional pre-training. This is another advantage of our proposed model.

5.2. Ablation Study

5.2.1. Modules Ablation

To evaluate the impact of each module on the performance of the proposed model, we conducted an ablation experiment on PathQuestions. We experimented with various combinations of the three modules to analyze their impacts. Table 4 shows the results.

As shown in the 2*rd* row in Table 4, when only the text encoder is used in the encoding stage, adding the guidance module can significantly improve the quality of the generated text. This is because the improved pointer network in the guidance module can avoid error-generated entities and out-of-distribution vocabulary and improve the fluency and accuracy of the generated text. However, due to the lack of a graph encoder, the knowledge of graph structure information is not fully accessible, and there is still room for improvement.

As shown in 3*rd* row in Table 4, when we add the graph encoder and eliminate the guidance module, the graph structure can be adequately captured and encoded, but the generation of improper nouns and out-of-distribution vocabulary leads to a particular gap between the generated text and the target result.

As shown in the 4*th* row in Table 4, when the dual-path encoder (consisting of the text encoder and the graph encoder) and the alignment module are used, the model can fully capture the knowledge graph structure information. There is also an improvement than the 3rd row in Table 4 on the quality of the generated text.

When the alignment module is not used (in 5*th* row in Table 4), the output of the dual-path encoder is directly summed up as the input of the decoder. In this case, the graph structure information and text information from the dual-path encoder cannot be well aligned and utilized. Compared the 5*th* row to the 6*th* in Table 4, we found that adding an alignment module can improve the performance effectively since the alignment module can make the representation from the dual-path encoder better fused and aligned.

In summary, the three modules proposed in this paper are complementary to each other. The proposed model can achieve excellent results.

5.2.2. Alignment Module Layers

To test the impact of the number of transformer layers in the alignment module, we experiment on PathQuestions. We set the number of layers of

	PathQuestions			WEBNLG(C)			WEBNLG(U)		
	B-4	**M**	**R-L**	**B-4**	**M**	**R-L**	**B-4**	**M**	**R-L**
NPT†	61.48	44.57	77.72	48.00	36.00	65.00	61.00	42.00	71.00
KGPT†	-	-	-	-	-	-	64.11	46.30	74.57
BART†	63.74	47.23	77.76	56.65	44.51	70.94	64.55	46.51	75.13
T5†	58.95	44.72	76.58	58.66	**46.04**	73.06	64.42	46.58	74.77
JointGT†	65.89	48.25	78.87	58.55	45.01	72.31	65.92	47.15	76.10
GAP‡	66.18	48.12	76.38	-	-	-	66.20	46.77	**76.36**
Ours	**67.20**	**48.56**	**79.62**	**59.33**	44.92	**73.14**	**66.41**	**47.38**	76.18

Table 3: Performance comparison on PathQuestions, WebNLG(U) and webNLG(C). The symbols † and ‡ indicate that the data are from JointGT(Ke et al., 2021) and GAP(Colas et al., 2022), respectively. **Bold** and underline fonts indicate the **best-performing** and second-best-performing results, respectively. We also used the abbreviations of the evaluation metrics: B-4 is BLEU-4, M is METEOR, and R-L is ROUGE-L.

Dual-path Encoder	Alignment Module	Guidance Module	BLEU-4	METEOR	ROUGE-L
only text encoder	×	×	65.89	48.25	78.87
only text encoder	×	✓	66.88	48.49	79.51
✓	×	×	66.01	48.15	78.81
✓	✓	×	66.64	48.35	79.23
✓	×	✓	66.25	48.33	79.02
✓	✓	✓	67.20	48.56	79.62

Table 4: Ablation study on PathQuestions

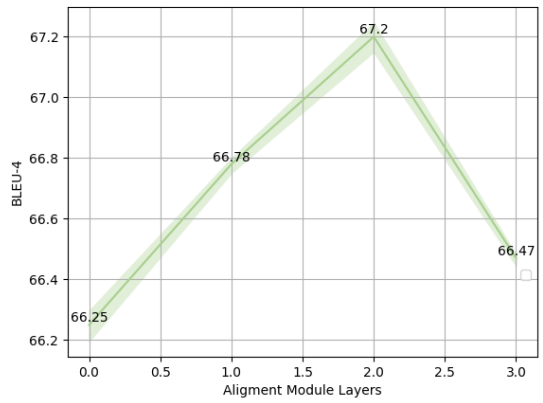

Figure 3: The impact of the transformer layers in the alignment module on PathQuestions.

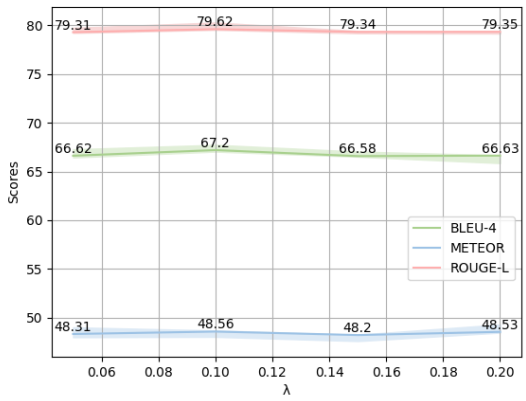

Figure 4: The effect of λ values on model performance on PathQuestions.

the alignment module $N \in [0, 3]$. The results are shown in Figure 3.

From Figure 3, we found that when the number of layers is 2, the proposed model achieves the best results, improving 1.31% on the BLEU score. The point with 0 layer represents the model without the alignment module. In such a case, the performance is the worst. The performance with a 3-layer transformer block is worse than that with a 2-layer transformer block since increasing the transformer layers will lead to overfitting. In the proposed model, a 2-layer transformer block in the alignment module is adequate to align and fuse the representations from the dual-path encoder.

5.2.3. Graph Encoder Weights

We use a graph encoder to capture the structure information of knowledge graphs. To explore the impact of the weight coefficient λ in Eq. 8 in the alignment module, we conduct experiments on PathQuestions. We set $\lambda \in \{0.05, 0.1, 0.15, 0.2\}$. Figure 4 shows the impact of λ.

As shown in Figure 4, the model achieves the best results when $\lambda = 0.1$. Figure 4 also indicates that the structural information affects the quality of generation.

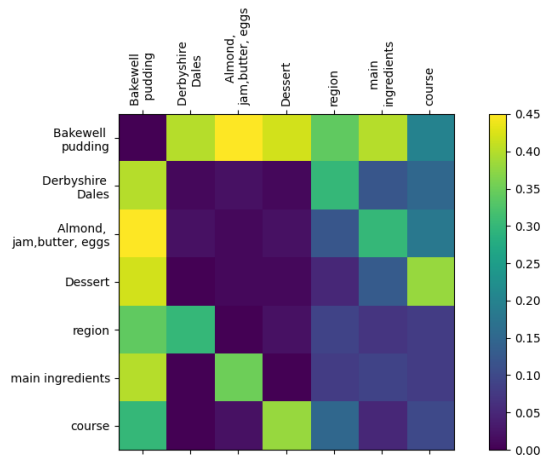

Figure 5: Visualization of the attention in the alignment module.

5.3. Attention in Alignment Module

In the alignment module, the attention represents the correlations between elements in the triples in knowledge graphs. We obtain the attention in the alignment module for the example in Figure 1 and visualize it in Figure 5. The attention between the elements in the same triples shows higher attention values than that between the elements not appearing in any same triples. This suggests that our model effectively captures the attention information in the knowledge graphs.

5.4. Case Study

To further show the effectiveness of the proposed approach, we list some examples of the generated texts on the WebNLG test set in Table 5. We can find that the generated texts are almost semantically equivalent to the reference texts. Since there are different ways to express the same semantics, the automatic evaluation metrics may give low scores to the results with the same semantics but different expressions.

6. Conclusion

In this paper, we propose a synergistic model of the dual-path encoder, alignment module, and guidance module for KG-to-text generation. The proposed model obtains a more competitive performance than the baselines. Another advantage of the proposed model is that it adapts the graph structure by the dual-path encoder and requires no additional pre-training. Through extensive experiments, we found that the dual-path encoder can encode the knowledge graph more effectively, and introducing a graph encoder can avoid the miss of structure information in the KG-to-text task. Meanwhile, using an alignment module is better than directly concatenating the graph representation and text representation for model performance. In addition, introducing the guidance module can avoid error-generated entities and ensure the fluency and accuracy of the generated text.

7. Acknowledgements

This work was funded by the National Natural Science Foundation of China (Grant No. 62366036), the National Education Science Planning Project (Grant No. BIX230343), Key R&D and Achievement Transformation Program of Inner Mongolia Autonomous Region (Grant No. 2022YFHH0077), The Central Government Fund for Promoting Local Scientific and Technological Development (Grant No. 2022ZY0198), Program for Young Talents of Science and Technology in Universities of Inner Mongolia Autonomous Region (Grant No. NJYT24033), Inner Mongolia Autonomous Region Science and Technology Planning Project (Grant No. 2023YFSH0017), Joint Fund of Scientific Research for the Public Hospitals of Inner Mongolia Academy of Medical Sciences (Grant No.2023GLLH0035), Natural Science Foundation of Inner Mongolia(Grant No. 2021BS06004).

8. Bibliographical References

Alfred V. Aho and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation and Compiling*, volume 1. Prentice-Hall, Englewood Cliffs, NJ.

American Psychological Association. 1983. *Publications Manual*. American Psychological Association, Washington, DC.

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853.

Galen Andrew and Jianfeng Gao. 2007. Scalable training of L_1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*, pages 33–40.

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, Ann Arbor, Michigan, USA, June 29, 2005*, pages 65–72. Association for Computational Linguistics.

Prediction	Akron, Ohio is 306 meters above sea level and has a population density of 1239.3 inhabitants per square kilometer. The total area of Akron is 161.54 square kilometers.
Reference	Akron, Ohio is at a height of 306.0 above sea level with a total area of 161.54 sq Km and it has a 1239.3 inhabitants per square km population density.
Prediction	Akeem Priestley was born in Jamaica which was led by Patrick Allen and Elizabeth II.
Reference	Jamaica, whose queen is Elizabeth II, is led by Patrick Allen and is the birthplace of Akeem Priestley.
Prediction	Aleksander Barkov, Jr., who weighs 96.1632 kilograms, was drafted by the Florida Panthers where Dale Tallon is the general manager.
Reference	Aleksandr Barkov Jr weighs 96.1632 kgs and his draft team is the Florida Panthers whose general manager is Dale Tallon.
Prediction	320 South Boston Building was completed in 1929 and has 22 floors and a height of 121.92 meters.
Reference	In 1929 the 320 South Boston Building was completed with 22 floors and a height of 121.92 metres.
Prediction	Construction of the Addis Ababa City Hall began in 1961 and finished in 1964 with a floor area of 140000.0 square meters.
Reference	Construction on Addis Ababa City Hall, which has a 140000.0 (square meters) floor area, began in 1961 and was completed in 1964.
Prediction	Asher and Mary Isabelle Richardson House in Asherton, Texas was built in 1911 and was added to the National Register of Historic Places on November 22nd, 1988.
Reference	Asher and Mary Isabelle Richardson House was constructed in 1911 in Asherton Texas and was added to the National Register of Historic Places, on 22nd November 1988.

Table 5: Examples of the generated texts from our model on the WebNLG test set.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Neural Information Processing Systems*.

BSI. 1973a. *Natural Fibre Twines*, 3rd edition. British Standards Institution, London. BS 2570.

BSI. 1973b. Natural fibre twines. BS 2570, British Standards Institution, London. 3rd. edn.

A. Castor and L. E. Pollux. 1992. The use of user modelling to guide inference and learning. *Applied Intelligence*, 2(1):37–53.

Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. 1981. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114–133.

Wenhu Chen, Yu Su, Xifeng Yan, and William Yang Wang. 2020. KGPT: knowledge-grounded pre-training for data-to-text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 8635–8648. Association for Computational Linguistics.

J.L. Chercheur. 1994. *Case-Based Reasoning*, 2nd edition. Morgan Kaufman Publishers, San Mateo, CA.

N. Chomsky. 1973. Conditions on transformations. In *A festschrift for Morris Halle*, New York. Holt, Rinehart & Winston.

Anthony Colas, Mehrdad Alvandipour, and Daisy Zhe Wang. 2022. GAP: A graph-aware language model framework for knowledge graph-to-text generation. In *Proceedings of the 29th International Conference on Computational Linguistics, COLING 2022, Gyeongju, Republic of Korea, October 12-17, 2022*, pages 5755–5769. International Committee on Computational Linguistics.

James W. Cooley and John W. Tukey. 1965. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19(90):297–301.

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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), pages 4171–4186. Association for Computational Linguistics.

Umberto Eco. 1990. *The Limits of Interpretation*. Indian University Press.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime G. Carbonell. 2016. Generation from abstract meaning representation using tree transducers. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 731–739. The Association for Computational Linguistics.

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, INLG 2017, Santiago de Compostela, Spain, September 4-7, 2017*, pages 124–133. Association for Computational Linguistics.

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. *CoRR*, abs/2006.04702.

Dan Gusfield. 1997. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, Cambridge, UK.

Paul Gerhard Hoel. 1971a. *Elementary Statistics*, 3rd edition. Wiley series in probability and mathematical statistics. Wiley, New York, Chichester. ISBN 0 471 40300.

Paul Gerhard Hoel. 1971b. *Elementary Statistics*, 3rd edition, Wiley series in probability and mathematical statistics, pages 19–33. Wiley, New York, Chichester. ISBN 0 471 40300.

Otto Jespersen. 1922. *Language: Its Nature, Development, and Origin*. Allen and Unwin.

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, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 2398–2409. International Committee on Computational Linguistics.

Pei Ke, Haozhe Ji, Yu Ran, Xin Cui, Liwei Wang, Linfeng Song, Xiaoyan Zhu, and Minlie Huang. 2021. Jointgt: Graph-text joint representation learning for text generation from knowledge graphs. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 2526–2538. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 2284–2293. Association for Computational Linguistics.

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, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. Association for Computational Linguistics.

Junyi Li, Tianyi Tang, Wayne Xin Zhao, Zhicheng Wei, Nicholas Jing Yuan, and Ji-Rong Wen. 2021. Few-shot knowledge graph-to-text generation with pretrained language models. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 1558–1568. Association for Computational Linguistics.

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

Jin Liu, Chongfeng Fan, Fengyu Zhou, and Huijuan Xu. 2022. Syntax controlled knowledge graph-to-text generation with order and semantic consistency. In *Findings of the Association for Computational Linguistics: NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages

1278–1291. Association for Computational Linguistics.

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 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 2267–2277. Association for Computational Linguistics.

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, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318. ACL.

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, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 43–54. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

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.

Mohammad Sadegh Rasooli and Joel R. Tetreault. 2015. Yara parser: A fast and accurate dependency parser. *Computing Research Repository*, arXiv:1503.06733. Version 2.

Leonardo F. R. Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. 2020. Investigating pretrained language models for graph-to-text generation. *CoRR*, abs/2007.08426.

Martin Schmitt, Sahand Sharifzadeh, Volker Tresp, and Hinrich Schütze. 2020. 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, EMNLP 2020, Online, November 16-20, 2020*, pages 7117–7130. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1073–1083. Association for Computational Linguistics.

Charles Joseph Singer, E. J. Holmyard, and A. R. Hall, editors. 1954–58. *A history of technology*. Oxford University Press, London. 5 vol.

Jannik Strötgen and Michael Gertz. 2012. Temporal tagging on different domains: Challenges, strategies, and gold standards. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, pages 3746–3753, Istanbul, Turkey. European Language Resource Association (ELRA).

S. Superman, B. Batman, C. Catwoman, and S. Spiderman. 2000. *Superheroes experiences with books*, 20th edition. The Phantom Editors Associates, Gotham City.

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, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1627–1637. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2017. Graph attention networks. *CoRR*, abs/1710.10903.

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.

Donghan Yu, Chenguang Zhu, Yiming Yang, and Michael Zeng. 2022. JAKET: joint pre-training of knowledge graph and language understanding. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 11630–11638. AAAI Press.

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, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 1441–1451. Association for Computational Linguistics.

9. Language Resource References

Gardent, Claire and Shimorina, Anastasia and Narayan, Shashi and Perez-Beltrachini, Laura. 2017. *Creating Training Corpora for NLG Micro-Planning*.

Mantong Zhou and Minlie Huang and Xiaoyan Zhu. 2018. *An Interpretable Reasoning Network for Multi-Relation Question Answering*. Association for Computational Linguistics.