GraSAME: Injecting Token-Level Structural Information to Pretrained Language Models via Graph-guided Self-Attention Mechanism

Shuzhou Yuan and **Michael Färber**
Karlsruhe Institute of Technology
Center for Scalable Data Analytics and Artificial Intelligence (ScaDS.AI)
TU Dresden
{shuzhou.yuan, michael.faerber}@kit.edu

Abstract

Pretrained Language Models (PLMs) benefit from external knowledge stored in graph structures for various downstream tasks. However, bridging the modality gap between graph structures and text remains a significant challenge. Traditional methods like linearizing graphs for PLMs lose vital graph connectivity, whereas Graph Neural Networks (GNNs) require cumbersome processes for integration into PLMs. In this work, we propose a novel graph-guided self-attention mechanism, GraSAME. GraSAME seamlessly incorporates token-level structural information into PLMs without necessitating additional alignment or concatenation efforts. As an end-to-end, lightweight multimodal module, GraSAME follows a multi-task learning strategy and effectively bridges the gap between graph and textual modalities, facilitating dynamic interactions between GNNs and PLMs. Our experiments on the graph-to-text generation task demonstrate that GraSAME outperforms baseline models and achieves results comparable to state-of-the-art (SOTA) models on WebNLG datasets. Furthermore, compared to SOTA models, GraSAME eliminates the need for extra pre-training tasks to adjust graph inputs and reduces the number of trainable parameters by over 100 million.

1 Introduction

The paradigm of pre-training and fine-tuning has increasingly become the standard approach for leveraging the inherent knowledge of language models in a wide range of Natural Language Processing (NLP) tasks (Xu et al., 2021). Pretrained Language Models (PLMs) like Transformer (Vaswani et al., 2017), T5 (Raffel et al., 2020), and GPT (Brown et al., 2020), which are trained on extensive text corpora, have demonstrated remarkable performance across various NLP challenges. However, these models primarily focus on textual data, presenting a significant limitation in processing structured information, such as Knowledge Graphs (KGs), molecular graph and social networks. Such graph structures are crucial for storing external knowledge and can significantly enhance PLM's performance on knowledge-driven tasks (Zhang et al., 2019; Peters et al., 2019). This limitation becomes particularly evident in tasks that require a deep understanding of both textual and structural data, such as graph-to-text generation (Gardent et al., 2017), KG-based fact checking (Kim et al., 2023), and translation between molecules and natural language (Edwards et al., 2022).

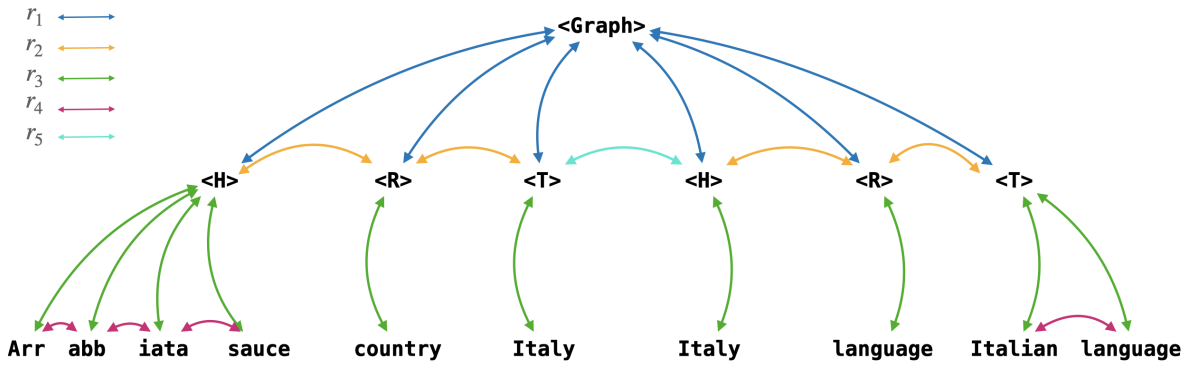
To address the challenge of processing structural input in PLMs, recent research has explored two main strategies: linearizing graph structures into text sequences (Harkous et al., 2020; Ribeiro et al., 2021a; Schmitt et al., 2021), and encoding structural information using Graph Neural Networks (GNNs) (Yao et al., 2020; Ribeiro et al., 2021b; Li et al., 2021; Zhang et al., 2022). While linearization allows direct fine-tuning of PLMs, studies have shown that it often fails to preserve the inherent structural information and explicit node connectivity (Song et al., 2018; Ribeiro et al., 2019). Conversely, while GNNs effectively encode complex structures, the modality difference between text and graphs complicates the integration with PLMs, requiring additional training for aligning and concatenating embeddings from graph and textual modality (Li et al., 2021; Zhang et al., 2022).

To merge the strengths of PLMs and GNNs, we introduce GraSAME, a novel **Gra**ph-guided **S**elf-**A**ttention **ME**chanism, enhancing PLMs' ability to process graph inputs. As depicted in Figure 1, we construct a token-level hierarchical graph structure from the linearized graph sequence to maintain the input graph's structural integrity. GraSAME is designed to learn the token-level graph information from a GNN and integrate it seamlessly into the text representation for PLM. Based on a standard transformer architecture, we substitute the

Findings of the Association for Computational Linguistics: NAACL 2024, pages 920–933
June 16-21, 2024 ©2024 Association for Computational Linguistics

Input text: Translate graph to English: **<Graph>** **<H>** Arrabbiata sauce **<R>** country **<T>** Italy **<H>** Italy **<R>** language **<T>** Italian language

Hierarchical graph:



Target text: Arrabbiata sauce can be found in Italy where the Italian language is spoken.

Figure 1: An example of KG-to-text generation. We visualize the hierarchical graph structure derived from the linearized graph input, tokenized by T5-large tokenizer. Each token from the input text is depicted as a node. Various relation types, each indicated by a unique color, are assigned between nodes to establish the hierarchical structure and ensure effective information flow among neighboring nodes.

self-attention layers in the encoder with GraSAME. As GraSAME effectively encodes the graph structure, we train only its parameters while keeping the PLM's parameters frozen. This approach enables PLMs equipped with GraSAME to simultaneously process structural and textual information dynamically, eliminating the need for complex alignment or concatenation of different modalities.

Applied to KG-to-text generation task WebNLG, we integrate GraSAME into the encoder-decoder model T5 with multi-task fine-tuning. The KG-to-text generation is particularly suitable as it necessitates the processing of both graph and text information, providing a clear intuition to assess the effectiveness of GraSAME. Our experiments demonstrate that GraSAME is compatible with various GNN architectures such as GraphSAGE (Hamilton et al., 2017), GAT (Veličković et al., 2018) and RGCN (Schlichtkrull et al., 2018), yielding performance that not only surpasses baseline models but also comparable to state-of-the-art (SOTA) models. Moreover, GraSAME effectively integrates structural information purely during fine-tuning and saves over 100 million trainable parameters.

In summary, our contributions are: **i)** Introducing a novel graph-guided attention mechanism GraSAME to incorporate explicit structural information into PLMs. This innovation enables PLMs to process both textual and structural inputs smoothly, bridging the modality gap of GNNs and PLMs. With GraSAME, PLMs can dynamically interact with GNNs, effectively interpreting graph inputs, which is crucial for NLP tasks that require structural information. **ii)** Applying GraSAME to KG-to-text generation on WebNLG datasets, achieving results comparable to SOTA models while saving over 100 million trainable parameters.

2 Related Work

Structural Information for PLMs. Although PLMs inherit linguistic structure information from pre-training (Nie et al., 2024), external structural information helps PLMs enhance their ability to understand the syntax of natural language (Yang et al., 2022), summarize source code (Choi et al., 2021) and generate better text (Song et al., 2020). Much initial focus of infusing structural information into PLMs has been on modifying pre-training objectives (Peters et al., 2019; Xiong et al., 2020; He et al., 2020). Zhang et al. (2019) utilized both textual corpora and KGs to pre-train an enhanced language representation model. Ke et al. (2021) proposed three new pre-training tasks to explicitly enhance the graph-text alignment. Also, recent efforts increasingly aimed at injecting structural information into PLMs during fine-tuning for various NLP tasks (Yasunaga et al., 2021; Ribeiro et al., 2021b). Wang et al. (2021) proposed K-Adapter to infuse knowledge into PLMs. Zhang et al. (2022)

came up with GreaseLM model to utilise KG information for question answering.

KG-to-text Generation. Previous approaches to enabling PLMs to process graph inputs often relied on linearizing the input graph into a text sequence (Harkous et al., 2020; Mager et al., 2020; Ribeiro et al., 2021b; Colas et al., 2022). Ribeiro et al. (2021a) investigated PLMs on graph-to-text generation using linearized graphs and found that this method is effective, yet results in the loss of specific edge connections in graphs (Song et al., 2018; Beck et al., 2018; Ribeiro et al., 2019). Also, Yuan and Faerber (2023) evaluated generative models using linearized graph and uncovered the issues of hallucinations in the models. An alternative approach to encode the graph inputs is leveraging GNNs (Koncel-Kedziorski et al., 2019; Ribeiro et al., 2020). But this method typically entails additional steps such as aligning modalities and concatenating embeddings (Li et al., 2021), which adds complexity to the development of a seamless end-to-end pipeline for integrating GNN with PLM. Diverging from the previous methods, our work synthesizes the strengths of both linearized graph and GNN. Moreover, GraSAME also follows a lightweight fine-tuning avoiding updating the parameters of the whole model, inspired by the adapter and parameter-efficient fine-tuning approaches (Houlsby et al., 2019; Ribeiro et al., 2021b; Wang et al., 2021; Yuan et al., 2024).

3 Model

In this section, we detail the components of our model. Theoretically, GraSAME is adaptable to any attention-based PLMs. We choose T5 model (Raffel et al., 2020) as our foundation due to its encoder-decoder architecture, which is well-suited for KG-to-text generation.

3.1 Encoder-Decoder Model

Encoder-decoder model, such as T5, is a classic Transformer model consisting of encoder and decoder layers. Each encoder layer includes two distinct sublayers: a self-attention mechanism and a position-wise fully connected feed-forward network. The self-attention mechanism utilizes h distinct attention heads. Consider a conditional generation task such as KG-to-text generation, where the input is a sequence of tokens $x = (x_1, \ldots, x_n)$ with each $x_i \in \mathbb{R}^{d_x}$, and the aim is to generate target sequence of tokens $y = (y_1, \ldots, y_n)$. The attention head processes an input sequence, the outputs of all attention heads are merged via concatenation, followed by a parameterized linear transformation to yield the final output of the self-attention sublayer. The computation of each output element z_i, with each $z_i \in \mathbb{R}^{d_z}$, involves a weighted sum of linearly transformed input elements, defined as:

$$z_i = \sum_{j=1}^{n} \alpha_{ij}(x_j W^V), \tag{1}$$

where α_{ij} represents the weight coefficient, calculated using a softmax function:

$$\alpha_{ij} = \text{softmax}(\frac{(x_i W^Q)(x_j W^K)^T}{\sqrt{d_k}}). \tag{2}$$

The matrices $W^V, W^Q, W^K \in \mathbb{R}^{d_x \times d_z}$ are layer-specific trainable parameters, and are distinct for each attention head.

3.2 Graph-guided Self-Attention Mechanism

Self-attention allows for the interaction of token representations by treating each input sequence as a fully-connected graph with tokens as nodes (Yao and Wan, 2020). However, this process does not retain the original structural information and explicit connectivity between the tokens. To address this issue, we introduce GraSAME, a method that integrates text with token-level hierarchical graph representation illustrated in Figure 1.

3.2.1 Architecture of GraSAME

GraSAME involves incorporating a GNN within the self-attention layer of the PLM. This addition enables the direct encoding of the hierarchical graph structure and facilitates the smooth transfer of structural information into the PLM. We visualize the architecture in Figure 2.

Graph Neural Network. The graph neural network models structural characteristics of the input graph effectively by using various graph convolutional layers. The primary goal of the GNN is to learn the representations for both individual nodes and the overall graph structure. In most GNN models, the node representation is updated iteratively by aggregating the representations of its neighboring nodes. The representation of node v_i at the l-th layer is represented by $h^{(l)}$, with the initial representation $h^{(0)}$ set to the node's feature vector x_i. The process of representation update at the l-th

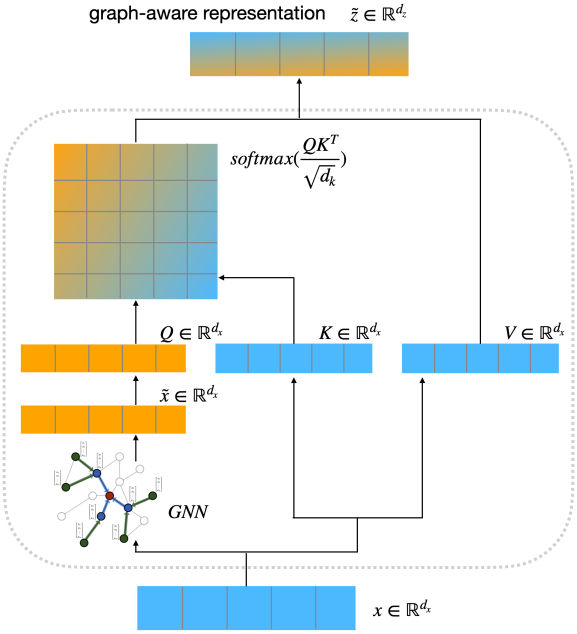


Figure 2: The architecture of GraSAME. The text embedding x is fed into a GNN along with the edge index derived from the hierarchical graph structure. This process generates a graph embedding $\tilde{x}$, which subsequently induces the Q vector. The Q vector then guides the self-attention mechanism to produce a graph-aware representation $\tilde{z}$. The visualization of GNN is taken from GraphSAGE (Hamilton et al., 2017).

layer involves two main operations:

$$a^{(l)} = \text{AGGREGATE}^{(l)}\left(\{h_j^{(l-1)} : j \in \mathcal{N}(v_i)\}\right), \quad (3)$$

$$h_i^{(l)} = \text{COMBINE}^{(l)}\left(h_i^{(l-1)}, a^{(l)}\right), \quad (4)$$

where $\mathcal{N}(v_i)$ denotes the neighbors of v_i. The AGGREGATE function compiles message passing from these neighbors, using techniques like MEAN, MAX, or SUM, varying with the GNN architecture. The COMBINE function then integrates this aggregated information into the node's current representation, thereby updating it.

Incorporating Method. Drawing inspiration from the multimodal self-attention layer by Yao and Wan (2020), we present a graph-guided self-attention architecture designed to simultaneously encode text representation and hierarchical graph structure. In our approach, all tokens in the input text are treated as nodes, with their initial features derived from the token representations in $h^{(l)}$. As shown in Figure 2, the token representations are aggregated and updated through a GNN layer. This process generates the vector Q, which subsequently guides the self-attention layer in the encoder of the PLM.

Formally, we adapt Equation 2 such that the weight coefficient $\tilde{\alpha_{ij}}$ is derived from the node representation $\tilde{x_i}$ in the graph modality, and the token representation x_j from the text modality:

$$\tilde{\alpha_{ij}} = \text{softmax}\left(\frac{(\tilde{x_i}W^Q)(x_jW^K)^T}{\sqrt{d_k}}\right). \quad (5)$$

The output of the self-attention layer is then calculated as:

$$\tilde{z_i} = \sum_{j=1}^{n} \tilde{\alpha_{ij}}(x_jW^V). \quad (6)$$

This modification ensures that the hidden word representations are influenced by the graph embedding. In each encoder layer of the model, we incorporate residual connections and layer normalization. The standard self-attention layer in the encoder is replaced with GraSAME, while the decoder retains the standard Transformer implementation. In the encoder's final layer, $\tilde{z_i}$ serves as the input to the decoder, which generates the target sequence.

3.2.2 Graph Representation

As PLMs are designed to process textual input only, it becomes necessary to perform certain preprocessing steps when addressing graph-based NLP tasks. Considering the task of KG-to-text generation, we represent the graph input as a linearized graph following prior studies (Harkous et al., 2020; Ribeiro et al., 2021a), and also extract a token-level hierarchical graph structure to ensure information flow among neighboring nodes.

Linearized Graph. In line with Ribeiro et al. (2021a), we linearize the graph into a sequence of text augmented with special tokens. As depicted in Figure 1, a KG triple is composed of a head, relation, and tail entity. Accordingly, we prepend each entity with special tokens: <H>, <R>, <T>. Furthermore, to distinguish between text and graph inputs, we introduce the <$Graph$> token.[1] Previous work (Ribeiro et al., 2021a) suggests that PLMs generate fluent text regardless of the linearization order of the graph. Hence we adhere to the default sequence in which triples appear in the dataset.

Hierarchical Graph Structure. We derive a token-level hierarchical graph structure from the linearized graph, as depicted in Figure 1. The

[1]For the KG-to-text generation task, the prompt "translate graph to English: " is added as a task description to match the input format of T5.

concept of this hierarchy is inspired by the tree-like structures observed in functional organizations such as companies, universities, and even insect societies (Pooler, 2017; Anderson et al., 2001), and has been explored in neural network research for knowledge representation as well (Ying et al., 2018; Moutsinas et al., 2021; Chen et al., 2020). In this structure, each token dynamically interacts within its hierarchical tier through message passing, which we believe helps preserve the graph's original explicit connectivity and ensures effective information flow among neighboring nodes. Formally, let's consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ comprises labeled edges of the form (u, r, v). Here, u and v represent nodes in $\mathcal{V}$, and r from $\mathcal{R}$ denotes the relation type. In our structure, each token in the linearized graph is considered as a node, and we define specific relation types r between two nodes to enhance the hierarchy:

1. r_1 connects the global node <$Graph$> to special tokens <H>, <R>, <T>.
2. r_2 links adjacent special tokens within a triple.
3. r_3 connects special tokens to their respective entity tokens.
4. r_4 joins consecutive tokens within entities.
5. r_5 associates special tokens sharing the same entity.

An example of such a hierarchical graph is presented in Figure 1. We design the edges to be bidirectional, as this approach of information propagation in multiple directions can enhance the model's performance (Yao et al., 2020).

4 Multi-Task Fine-Tuning

Our training approach employs a multi-task learning strategy with efficient, lightweight fine-tuning. We initialize the model with pretrained parameters denoted by ϕ. The parameters of the PLM remain frozen, and only the GNN component within GraSAME is updated, given its effective encoding of graph structure.

4.1 Training Objectives

We retain the standard language model objective of generating the next token in a sequence while introducing an additional graph reconstruction task. This task is designed to strengthen the relation types between pairs of nodes, enhancing the hierarchy.

Text Generation. The text generation task is implemented by a PLM with a language modeling head on top. Given an input sequence x and a graph representation $\mathcal{G}$, the model aims to generate a target sequence y by minimizing the cross-entropy loss:

$$\mathcal{L}_{TG} = -\sum_{i=1}^{|y|} \log P_\phi(y_i | y_{1:i-1}, x, \mathcal{G}), \quad (7)$$

where P_ϕ is the generative probability from PLM.

Graph Reconstruction. Building on previous work that focused on predicting relationships between entities (Song et al., 2020; Li et al., 2021), our approach reformulates the graph reconstruction task. We aim to predict the relation type r in the triple (u, r, v), where u and v are nodes in the hierarchical graph structure. Node representations h_u and h_v are derived from the last hidden states of the PLM's encoder. Consequently, the probability of relation r is given by:

$$p(r|u, v) = \text{softmax}(W[h_u; h_v] + b), \quad (8)$$

where W and b are trainable parameters. The loss for graph reconstruction is computed using cross-entropy loss:

$$\mathcal{L}_{GR} = -\sum_{\langle u,r,v\rangle \in \mathcal{E}} \log p(r|u, v). \quad (9)$$

We integrate the text generation loss and the graph reconstruction loss to train the PLM. The overall training loss is defined as follows:

$$\mathcal{L}_{total} = \mathcal{L}_{TG} + \lambda \mathcal{L}_{GR}, \quad (10)$$

where λ is a weighting coefficient.[2]

5 Experiments

In this section, we introduce the details of our experiments on KG-to-text generation task. We modify T5 for conditional generation from Huggingface (Wolf et al., 2019), and implement GraSAME with the GNN layers provided by PyTorch Geometric (Fey and Lenssen, 2019).

[2]We observed that the value of λ significantly impacts performance. After tuning on the validation set of the *WebNLG unconstrained* dataset, we set λ to 0.08, which yielded the best BLEU score. Further details are provided in Appendix C.

5.1 Dataset

WebNLG[3] (Gardent et al., 2017) is a commonly-used benchmark in KG-to-text generation (Chen et al., 2020; Li et al., 2021; Li and Liang, 2021; Colas et al., 2022; Ke et al., 2021). We employ WebNLG version 2.1 for our experiments, as it represents a refined version of the widely-used version 2.0 (Shimorina and Gardent, 2018; Chen et al., 2020; Ke et al., 2021; Colas et al., 2022). This version offers two distinct splits: *unconstrained* and *constrained*.

WebNLG unconstrained. Each WebNLG sample comprises several KG triples and a corresponding descriptive text. The triples are structured as *(head, relation, tail)* and the model is supposed to generate fluent text to describe the input triples. An illustrative example is presented in Figure 1. In the *unconstrained* dataset, there is no overlap of input graphs between the train, validation, and test sets.

WebNLG constrained. The data structure of *constrained* is as same as *unconstrained* dataset. However, the *constrained* dataset presents a greater challenge by ensuring that there is no overlap of triples in the input graphs across train, validation and test sets.

5.2 Setting

As an enhancement of self-attention mechanism, a foundational model is required for the implementation of GraSAME. For our experiments, T5-large serves as the foundational model. Prior to training, we expand T5's vocabulary to include the special tokens <H>, <R>, <T>, and <$Graph$>. To ensure fair comparisons, we maintain consistent hyper-parameters across both the baseline and our models.[4] All models are fine-tuned with the training set. The BLEU score on validation set is employed to identify the best-performing model, which is subsequently evaluated on the test set.

Evaluation Metrics. To evaluate the performance of the models, we use the automatic evaluation metrics BLEU (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2014), ROUGE-L (Lin, 2004) and chrF++ (Popović, 2015) following previous work (Shimorina and Gardent, 2018; Ribeiro et al., 2021a). The evaluations are conducted using the official evaluation script from the WebNLG challenge (Shimorina and Gardent, 2018).

Baseline. T5-large is employed as the baseline model. Previous research (Ribeiro et al., 2021a; Ke et al., 2021) has demonstrated T5's SOTA performance on graph-to-text generation, making it a robust baseline (Clive et al., 2022). This choice allows for a fair comparison with our approach.

GraSAME is integrated into the T5-large model for our experiments. We investigate the efficacy of different GNNs for encoding the hierarchical graph structure and denote them as:

GraSAME-GAT. Graph Attention Network (GAT) (Veličković et al., 2018) is used as the GNN component in GraSAME. GAT utilizes an attention mechanisms to aggregate the information from neighbouring nodes.

GraSAME-RGCN. We also integrate Relational Graph Convolutional Network (RGCN) (Schlichtkrull et al., 2018) with GraSAME. RGCN extends the Graph Convolutional Network (Kipf and Welling, 2016), enabling it to process local graph neighborhoods within large-scale relational data.

GraSAME-SAGE. GraphSAGE (Hamilton et al., 2017) is an inductive framework that efficiently generates node embeddings for unseen nodes by leveraging node feature information. We also combine it with GraSAME.

5.3 Evaluation Results

5.3.1 Main Results

Model	A	P	B	M	R	C
GCN	✗	-	60.80	42.76	71.13	-
KGPT	✓	177M	64.11	46.30	74.57	-
JointGT(T5)	✓	265M	<u>66.14</u>	<u>47.25</u>	<u>75.91</u>	-
T5-large	✗	737M	61.41	45.96	71.70	75.27
GraSAME-GAT	✗	75.7M	60.44	44.91	70.73	72.49
GraSAME-RGCN	✗	453M	60.26	44.46	70.93	71.88
GraSAME-SAGE	✗	151M	**65.55**	**48.38**	**74.55**	**77.34**

Table 1: Results on *WebNLG unconstrained*. A denotes if additional pre-training tasks are implemented or not. P = Trainable parameters, B = BLEU, M = METEOR, R = ROUGE, C = chrF++. The results of GCN, KGPT, JointGT(T5) are re-printed from Shimorina and Gardent (2018), Chen et al. (2020) and Ke et al. (2021), respectively. **Bold** indicates the best score of the models we trained. <u>Underline</u> indicates the best score of SOTA models in previous work.

We present the primary results for *WebNLG unconstrained* in Table 1. Remarkably, our leading model, GraSAME-SAGE, surpasses the baseline T5-large in all metrics, despite having over 500 million fewer trainable parameters. We believe this

[3] https://synalp.gitlabpages.inria.fr/webnlg-challenge/

[4] Details on the hyper-parameters are provided in Appendix A.

Model	1 triple	2 triples	3 triples	4 triples	5 triples	6 triples	7 triples
T5	46.81	67.58	64.39	64.86	58.73	68.91	62.97
GraSAME	+7.80	+1.85	+3.05	+2.52	+3.73	-2.83	+8.61

Table 2: BLEU scores for different graph sizes in *WebNLG unconstrained* test set. T5 is the baseline model T5-large, GraSAME is the best performed model GraSAME-SAGE. +, - denote the difference of scores.

is due to the powerful encoding ability of GraphSAGE for unseen nodes (for example, the special tokens we add to the model's vocabulary manually). Although other models don't outperform the baseline T5, they still achieve noteworthy performance, achieving BLEU scores of over 60 with much fewer trainable parameters than baseline.

For comparison, we also include the results of the SOTA models from related work. Notably, GraSAME-SAGE outperforms GCN and KGPT on BLEU and METEOR scores, and achieves performance comparable to JoinGT(T5). Our METEOR score of 48.38 is even higher than that of JoinGT(T5). It is worth mentioning that both KGPT and JointGT are pre-trained with additional tasks to fine-tune KG-to-text generation. Additionally, JointGT(T5) has 114 million more trainable parameters than GraSAME-SAGE. This highlights the efficiency of our approach, demonstrating that GraSAME can enhance PLMs by leveraging token-level structural information for KG-to-text generation.

The results for *WebNLG constrained* are similar as for *WebNLG unconstrained*. As illustrated in Table 3, GraSAME-SAGE outperforms the baseline T5, while our other models achieve comparable results to it with higher BLEU scores and fewer trainable parameters. In comparison to JoinGT(T5), GraSAME-SAGE maintains over 98% performance with fewer than 114 million trainable parameters, and it doesn't require additional pre-training tasks. This prove the generalization of our method under the constrained condition, where there is no overlap between training and test triples.

5.3.2 Detailed Analysis on Graph Size

We conduct a comprehensive analysis focusing on input graph size, as detailed in Table 2. Notably, GraSAME consistently improves the BLEU scores across various input triple counts, except for six-triple inputs. The most pronounced improvement occurs with seven-triple inputs, where the BLEU score surpasses the baseline by 8.61. Seven-triple inputs form the most complex graph structure in the test set, highlighting the efficacy of GraSAME in fortifying the PLM's capacity to process complex graph inputs through token-level structural integration. Interestingly, a significant improvement also emerges in one-triple input graphs, likely due to the baseline model's propensity for generating hallucinations with shorter input graphs.[5]

Model	A	P	B	M	R	C
JointGT(T5)	✓	265M	61.01	46.32	73.57	-
T5-large	✗	737M	58.77	**46.12**	72.01	73.22
GraSAME-GAT	✗	75.7M	59.21	45.38	71.01	72.79
GraSAME-RGCN	✗	453M	60.13	45.47	71.79	72.88
GraSAME-SAGE	✗	151M	**60.27**	45.81	**72.01**	**73.29**

Table 3: Results on *WebNLG constrained*. A denotes if additional pre-training tasks are implemented or not. P = Trainable parameters, B = BLEU, M = METEOR, R = ROUGE, C = chrF++. The results of JointGT(T5) are re-printed from Ke et al. (2021). **Bold** indicates the best score of the models we trained.

5.4 Human Evaluation

Model	Fluency	Meaning
Gold	5.59	5.71
T5	5.57	5.41
GraSAME	5.56	5.62

Table 4: Human evaluation on *WebNLG unstrained*. T5 denotes the baseline model T5-large, GraSAME denotes GraSAME-SAGE. The Fleiss' Kappa κ is 0.42, which indicates moderate agreement.

To further assess the quality of the generated text, we conduct a human evaluation using the crowdsourcing platform Amazon Mechanical Turk.[6] We randomly select 100 texts generated by both baseline and GraSAME-SAGE models, along with their corresponding gold standard references. In line with previous studies (Castro Ferreira et al., 2019; Ribeiro et al., 2021a), we ask three annotators to rate the texts on a 1-7 Likert scale across two dimensions: (i) Fluency: Assessing whether the text

[5] A more in-depth error analysis is presented in Section 6.

[6] https://www.mturk.com

is fluent, natural, and easy to read. (ii) Meaning: Evaluating if the text accurately conveys the information from the input graph without including extraneous information (hallucination). We specifically instruct annotators to pay close attention to instances of hallucination, as this issue has gained significant attention in recent PLM research (González Corbelle et al., 2022; Ji et al., 2023; Yuan and Faerber, 2023).

As indicated in Table 4, both the baseline and GraSAME models produce fluent text, scoring only marginally lower than the reference by 0.02 and 0.03, respectively. Regarding the meaningfulness of the generated text, GraSAME surpasses the baseline, achieving a score that is 0.21 points higher. This human evaluation confirms that GraSAME is capable of generating not only fluent text but also text that more accurately encapsulates the input information, while minimizing hallucinations.

5.5 Ablation Study

Model	BLEU	METEOR
GraSAME-SAGE	65.55	48.38
- bidirectional edges	60.74	45.13
- graph reconstruction	61.75	45.65

Table 5: Ablation study on *WebNLG unconstrained*.

We conduct an ablation study focusing on two key aspects of the model: the bidirectional edges and the graph reconstruction task. This study is implemented using the top-performing GraSAME-SAGE model on *WebNLG unconstrained*.

Bidirectional Edges: We retain a single edge direction in the hierarchical graph structure, specifically from bottom to top tokens.

Graph Reconstruction: We omit the graph construction loss during training, allowing the model to update solely based on the text generation loss.

The outcomes of the ablation study are detailed in Table 5. Both the bidirectional edge and graph reconstruction components significantly enhance the performance of GraSAME. Excluding either element results in a decrease in both BLEU and METEOR scores, with a marginally greater reduction observed upon the removal of bidirectional edges. This suggests that bidirectional edges are crucial for adequate message passing within the hierarchical graph structure.

Model	BLEU	METEOR	ROUGE	chrF++
Variation 1	60.62	46.57	71.03	74.67
Variation 2	60.07	46.01	70.17	74.08
GraSAME	65.55	48.38	74.55	77.34

Table 6: Results of model variations on *WebNLG unconstrained*. GraSAME = GraSAME-SAGE.

5.6 Model Variations

Considering the method of incorporating GNN into the self-attention layer, we introduce two additional variations of GraSAME, as visualized in Figure 4 of Appendix D. In Variation 1, the GNN generates graph embeddings to influence the vectors K and V, instead of Q. In Variation 2, we insert the GNN layer before the entire self-attention mechanism, which means the vectors K, V and Q are all derived from the graph embedding.

The results for the two variations are presented in Table 6. While Variation 1 outperforms Variation 2, both variations show a decrease in performance compared to the original GraSAME. This demonstrates that our proposed GraSAME is a valid and effective mechanism for encoding structural information.

6 Error Analysis

We illustrate three examples of generated text in Table 7. The first two examples have relatively short input texts, whereas the third example includes a longer and more complex input graph.

In the first and second examples, the text generated by T5 includes hallucinations not present in the input data. In contrast, GraSAME strictly adheres to facts conveyed by the input graphs. When handling more complex graph inputs, both T5 and GraSAME produce fluent and accurate text without introducing extraneous information. However, T5 makes a minor grammatical error with a preposition. This indicates that GraSAME effectively mitigates the issue of hallucinations, particularly with short and simple inputs. Moreover, for longer and more complex inputs, GraSAME demonstrates a superior understanding of the input structure, resulting in higher-quality text generation.

7 Conclusion

In this work, we introduce GraSAME, a novel graph-guided self-attention mechanism that enables PLMs to process token-level structural information. With GraSAME, PLMs are capable of

Model	Example
Input	translate graph to English: <*Graph*> <*H*> Iraq <*R*> language <*T*> Arabic
Reference	Iraq language is Arabic.
T5	Arabic is a language spoken in the country of Iraq. The country is a member of the United Nations.
GraSAME	Arabic is a language spoken in Iraq.
Input	translate graph to English: <*Graph*> <*H*> Madrid <*R*> country <*T*> Spain
Reference	Madrid is in the country of Spain.
T5	Madrid is a city in the country of Spain and is a popular tourist destination there.
GraSAME	Madrid is a city in the country of Spain.
Input	translate graph to English: <*Graph*> <*H*> Monocacy National Battlefield <*R*> location <*T*> Frederick County, Maryland <*H*> 14th New Jersey Volunteer Infantry Monument <*R*> established <*T*> "1907-07-11" <*H*> 14th New Jersey Volunteer Infantry Monument <*R*> country <*T*> "United States" <*H*> 14th New Jersey Volunteer Infantry Monument <*R*> category <*T*> Historic districts in the United States <*H*> 14th New Jersey Volunteer Infantry Monument <*R*> district <*T*> Monocacy National Battlefield <*H*> 14th New Jersey Volunteer Infantry Monument <*R*> state <*T*> "Maryland"
Reference	The 14th New Jersey Volunteer Infantry Monument is located on the Monocacy National Battlefield, Frederick County, Maryland. The monument was established in 1907-07-11 and is categorised as a historic district in the United States.
T5	The 14th New Jersey Volunteer Infantry Monument is located in the Monocacy National Battlefield, Frederick County, Maryland. It was established on 11 July 1907 and is categorised as a historic district in the United States.
GraSAME	The 14th New Jersey Volunteer Infantry Monument is located on the Monocacy National Battlefield in Frederick County, Maryland, United States. It was established on 11 July 1907 and is categorised as a historic district in the United States.

Table 7: Examples of text generated by T5-large and GraSAME-SAGE, with hallucinations highlighted in blue and other incorrect text marked in red.

handling text and graph input simultaneously. This approach facilitates seamless information flow between text and graph embeddings, eliminating the need for additional concatenation.

Evaluated on the KG-to-text generation task, GraSAME demonstrates performance comparable to SOTA models with significantly fewer trainable parameters. Through a detailed analysis of graph size and human evaluation, GraSAME demonstrates its enhanced ability to process more complex graph inputs and generate more accurate text. Moving forward, we aim to explore GraSAME's potential in encoding specific graph structures, like molecular graph (Edwards et al., 2022), in combination with large language models.

Limitation

Despite the effectiveness of our approach, we acknowledge several limitations in our work. Firstly, our method involves extracting a hierarchical graph structure from a linearized graph. While this structure facilitates efficient information exchange, it requires specific adjustments when applied to different datasets or tasks. Our goal is to combine the advantages of PLM and GNN, yet crafting a universal template that addresses all related tasks remains challenging.

Secondly, we observe that the training process of GraSAME is fast, but it tends to converge slower than the baseline model without GraSAME. This slower convergence is attributed to the GNN component of GraSAME not being pre-trained, necessitating additional training epochs for optimal interaction with the PLM.

Thirdly, our approach still incorporates the linearized graph as part of the input, which does not align with the pre-training process of PLMs typically conducted with plain text corpora in natural language. This misalignment could potentially lead to the forgetting of pre-trained knowledge. Addressing these limitations will be the focus of our future work.

Ethics Statement

This research was conducted in accordance with the ACM Code of Ethics. The datasets we used

are publicly available (Gardent et al., 2017; Kim et al., 2023), and we only used them to evaluate our models. We are not responsible for any potentially erroneous statements within the datasets. Additionally, care should be taken with the potential issues of hallucination when using the baseline model T5 for text generation.

Acknowledgements

We want to thank the anonymous reviewers for their helpful comments. Also, we would like to thank our colleague Yindong Wang for the inspiration of the idea, and Ercong Nie for his feedback on this work.

References

Carl Anderson, Nigel R Franks, and Daniel W McShea. 2001. The complexity and hierarchical structure of tasks in insect societies. *Animal Behaviour*, 62(4):643–651.

Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-sequence learning using gated graph neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 273–283, Melbourne, Australia. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

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 (EMNLP-IJCNLP)*, pages 552–562, Hong Kong, China. Association for Computational Linguistics.

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)*, pages 8635–8648, Online. Association for Computational Linguistics.

YunSeok Choi, JinYeong Bak, CheolWon Na, and Jee-Hyong Lee. 2021. Learning sequential and structural information for source code summarization. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2842–2851, Online. Association for Computational Linguistics.

Jordan Clive, Kris Cao, and Marek Rei. 2022. Control prefixes for parameter-efficient text generation. In *Proceedings of the 2nd Workshop on Natural Language Generation, Evaluation, and Metrics (GEM)*, pages 363–382, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

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*, pages 5755–5769, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 376–380, Baltimore, Maryland, USA. Association for Computational Linguistics.

Carl Edwards, Tuan Lai, Kevin Ros, Garrett Honke, Kyunghyun Cho, and Heng Ji. 2022. Translation between molecules and natural language. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 375–413, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Matthias Fey and Jan E. Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

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, Santiago de Compostela, Spain. Association for Computational Linguistics.

Javier González Corbelle, Alberto Bugarín-Diz, Jose Alonso-Moral, and Juan Taboada. 2022. Dealing with hallucination and omission in neural natural language generation: A use case on meteorology. In *Proceedings of the 15th International Conference on Natural Language Generation*, pages 121–130, Waterville, Maine, USA and virtual meeting. Association for Computational Linguistics.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

Hamza Harkous, Isabel Groves, and Amir Saffari. 2020. Have your text and use it too! end-to-end neural data-to-text generation with semantic fidelity. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2410–2424, Barcelona,

Spain (Online). International Committee on Computational Linguistics.

Bin He, Di Zhou, Jinghui Xiao, Xin Jiang, Qun Liu, Nicholas Jing Yuan, and Tong Xu. 2020. BERT-MK: Integrating graph contextualized knowledge into pre-trained language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2281–2290, Online. Association for Computational Linguistics.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

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*, pages 2526–2538, Online. Association for Computational Linguistics.

Jiho Kim, Sungjin Park, Yeonsu Kwon, Yohan Jo, James Thorne, and Edward Choi. 2023. FactKG: Fact verification via reasoning on knowledge graphs. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16190–16206, Toronto, Canada. Association for Computational Linguistics.

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

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, Volume 1 (Long and Short Papers)*, pages 2284–2293, Minneapolis, Minnesota. 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 pre-trained language models. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1558–1568, Online. Association for Computational Linguistics.

Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, et al. 2020. Pytorch distributed: Experiences on accelerating data parallel training. *Proceedings of the VLDB Endowment*, 13(12).

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

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, Online. Association for Computational Linguistics.

Giannis Moutsinas, Choudhry Shuaib, Weisi Guo, and Stephen Jarvis. 2021. Graph hierarchy: a novel framework to analyse hierarchical structures in complex networks. *Scientific Reports*, 11(1):13943.

Ercong Nie, Shuzhou Yuan, Bolei Ma, Helmut Schmid, Michael Färber, Frauke Kreuter, and Hinrich Schütze. 2024. Decomposed prompting: Unveiling multilingual linguistic structure knowledge in english-centric large language models. *arXiv preprint arXiv:2402.18397*.

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, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Robert Logan, 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)*, pages 43–54, Hong Kong, China. Association for Computational Linguistics.

James Pooler. 2017. *Hierarchical Organization in Society*. Routledge.

Maja Popović. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal. Association for Computational Linguistics.

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. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Leonardo F. R. Ribeiro, Claire Gardent, and Iryna Gurevych. 2019. Enhancing AMR-to-text generation with dual graph 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)*, pages 3183–3194, Hong Kong, China. Association for Computational Linguistics.

Leonardo F. R. Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. 2021a. Investigating pretrained language models for graph-to-text generation. In *Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI*, pages 211–227, Online. Association for Computational Linguistics.

Leonardo F. R. Ribeiro, Yue Zhang, Claire Gardent, and Iryna Gurevych. 2020. Modeling global and local node contexts for text generation from knowledge graphs. *Transactions of the Association for Computational Linguistics*, 8:589–604.

Leonardo F. R. Ribeiro, Yue Zhang, and Iryna Gurevych. 2021b. Structural adapters in pretrained language models for AMR-to-Text generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4269–4282, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*, pages 593–607. Springer.

Martin Schmitt, Leonardo F. R. Ribeiro, Philipp Dufter, Iryna Gurevych, and Hinrich Schütze. 2021. Modeling graph structure via relative position for text generation from knowledge graphs. In *Proceedings of the Fifteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-15)*, pages 10–21, Mexico City, Mexico. Association for Computational Linguistics.

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, Tilburg University, The Netherlands. Association for Computational Linguistics.

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, Online. Association for Computational Linguistics.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. A graph-to-sequence model for AMR-to-text generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1616–1626, Melbourne, Australia. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *International Conference on Learning Representations*. Accepted as poster.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Jianshu Ji, Guihong Cao, Daxin Jiang, and Ming Zhou. 2021. K-Adapter: Infusing Knowledge into Pre-Trained Models with Adapters. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1405–1418, Online. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Wenhan Xiong, Jingfei Du, William Yang Wang, and Veselin Stoyanov. 2020. Pretrained encyclopedia: Weakly supervised knowledge-pretrained language model. *ICLR*.

Dongkuan Xu, Ian En-Hsu Yen, Jinxi Zhao, and Zhibin Xiao. 2021. Rethinking network pruning – under the pre-train and fine-tune paradigm. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2376–2382, Online. Association for Computational Linguistics.

Erguang Yang, Chenglin Bai, Deyi Xiong, Yujie Zhang, Yao Meng, Jinan Xu, and Yufeng Chen. 2022. Learning structural information for syntax-controlled paraphrase generation. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2079–2090, Seattle, United States. Association for Computational Linguistics.

Shaowei Yao and Xiaojun Wan. 2020. Multimodal transformer for multimodal machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4346–4350, Online. Association for Computational Linguistics.

Shaowei Yao, Tianming Wang, and Xiaojun Wan. 2020. Heterogeneous graph transformer for graph-to-sequence learning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7145–7154, Online. Association for Computational Linguistics.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. QA-GNN: Reasoning with language models and knowledge graphs for question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 535–546, Online. Association for Computational Linguistics.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31.

Shuzhou Yuan and Michael Faerber. 2023. Evaluating generative models for graph-to-text generation. In *Proceedings of the 14th International Conference on Recent Advances in Natural Language Processing*, pages 1256–1264, Varna, Bulgaria. INCOMA Ltd., Shoumen, Bulgaria.

Shuzhou Yuan, Ercong Nie, Michael Färber, Helmut Schmid, and Hinrich Schütze. 2024. Gnnavi: Navigating the information flow in large language models by graph neural network. *arXiv preprint arXiv:2402.11709*.

X Zhang, A Bosselut, M Yasunaga, H Ren, P Liang, C Manning, and J Leskovec. 2022. Greaselm: Graph reasoning enhanced language models for question answering. In *International Conference on Representation Learning (ICLR)*.

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 Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451, Florence, Italy. Association for Computational Linguistics.

A Hyperparameters

We present the hyperparameters for T5-large and GraSAME in Table 8. We train the model until the training process converges. To keep a fair comparison, we keep the hyperparameters as same as possible. The learning rate of GraSAME is set larger than it for T5, because GraSAME converges slower due to fewer trainable parameters. The models are trained with 4 NVIDIA A100-SXM4-40GB GPUs.[7]

[7]We noted that when employing Distributed Data Parallel (Li et al., 2020), there is a possibility of sample duplication on the GPU to achieve the desired batch size, which could potentially impact the test set results. Consequently, we conducted the model evaluation on the test set using a single GPU for accuracy.

Hyperparameter	**T5**	**GraSAME**
`learning_rate`	3e-5	5e-5
`optimizer`	Adam	Adam
`batch_size`	10	10
`max_sequence_length`	187	187
`max_target_length`	120	120
`num_beam_search`	3	3
`random_seed`	123	123

Table 8: Hyperparameters.

B Data Statistics

We report the statistics of WebNLG in Table 9, which is the original split in WebNLG version 2.1.

Dataset	Size		
	| Train |	| Dev |	| Test |
WebNLG U	12876	1619	1600
WebNLG C	12895	1594	1606

Table 9: The statistics of dataset. WebNLG U = *WebNLG unconstrained*, WebNLG C = *WebNLG constrained*.

C Impact of Graph Reconstruction Loss

To investigate how graph reconstruction loss affect the performance and to determine the optimal value of λ in Equation 10, we visualize the tuning process in Figure 3. We use GraSAME-SAGE and the validation set of the *WebNLG unconstrained* dataset. The identified best value for λ is 0.08.

D Visualization of the Model Variations

To provide a clearer understanding of the model variations, we visualize the internal structure of the self-attention layer for the two model variations in Figure 4.

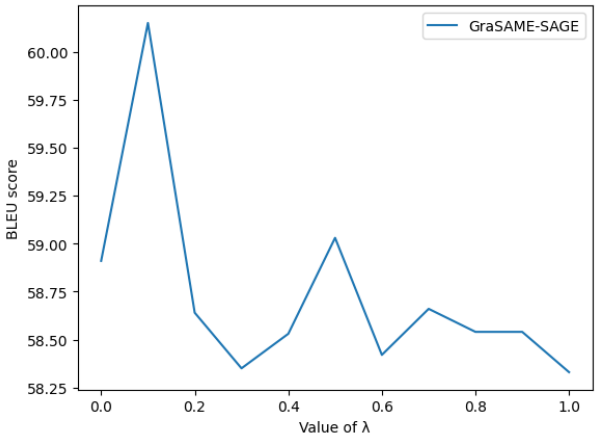
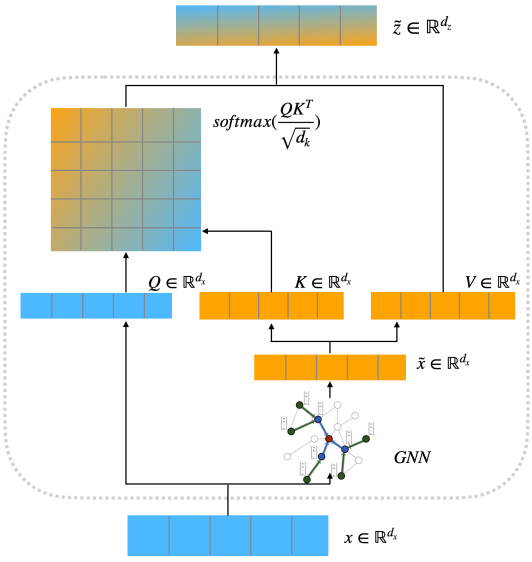
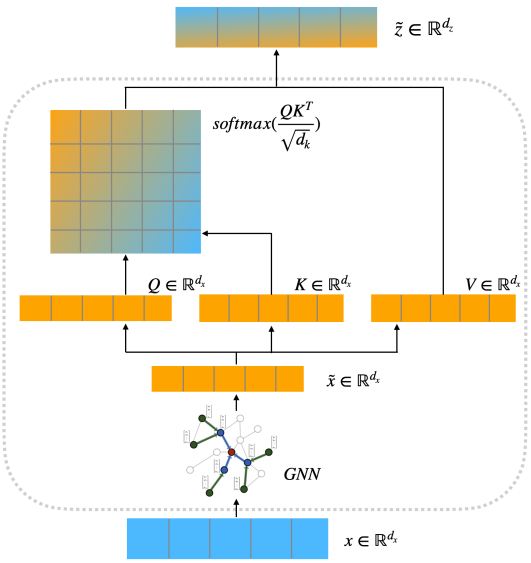


Figure 3: The impact of λ, experimented with GraSAME-SAGE on the validation set of *WebNLG unconstrained*.



(a) Variation 1: The text embedding x is fed into a GNN along with the edge index derived from the hierarchical graph structure. This process generates a graph embedding $\tilde{x}$, which subsequently induces the K and V vector. The Q vector then interacts with K and V vector to produce a graph-aware representation $\tilde{z}$.



(b) Variation 2: The text embedding x is fed into a GNN along with the edge index derived from the hierarchical graph structure. This process generates a graph embedding $\tilde{x}$, which subsequently induces the Q, K and V vector.

Figure 4: Visualization of two variations of GraSAME.