GAP: A Graph-aware Language Model Framework for Knowledge Graph-to-Text Generation

Anthony Colas* and **Mehrdad Alvandipour*** and **Daisy Zhe Wang**
Department of Computer Science, University of Florida
{acolas1, m.alvandipour, daisyw}@ufl.edu

Abstract

Recent improvements in KG-to-text generation are due to additional auxiliary pre-trained tasks designed to give the fine-tune task a boost in performance. These tasks require extensive computational resources while only suggesting marginal improvements. Here, we demonstrate that by fusing graph-aware elements into existing pre-trained language models, we are able to outperform state-of-the-art models and close the gap imposed by additional pre-train tasks. We do so by proposing a mask structure to capture neighborhood information and a novel type encoder that adds a bias to the graph-attention weights depending on the connection type. Experiments on two KG-to-text benchmark datasets show these models to be superior in quality while involving fewer parameters and no additional pre-trained tasks. By formulating the problem as a framework, we can interchange the various proposed components and begin interpreting KG-to-text generative models based on the topological and type information found in a graph.

1 Introduction

Due to the amount of data stored in Knowledge Graphs (KGs) (Auer et al., 2007; Vrandečić and Krötzsch, 2014; Bollacker et al., 2008; Yates et al., 2007; Bodenreider, 2004; Wishart et al., 2018), they are important to properly transcribe into natural language sentences, making them more easily comprehensible to a larger audience. This task, termed KG-to-text, has found recent success in generating knowledge-grounded dialog responses (Wen et al., 2016; Zhou et al., 2018), question answering (He et al., 2017; Bhowmik and de Melo, 2018; Pal et al., 2019; Agarwal et al., 2021), story generation (Guan et al., 2019; Ji et al., 2020), and event narration (Colas et al., 2021). KG-to-text involves encoding a KG, with its structure, in order to generate a coherent and representative textual description of the KG as shown in Figure 1. Thus, it is important to carefully consider the graph's structure when encoding the input data in order to properly generate its textual description.

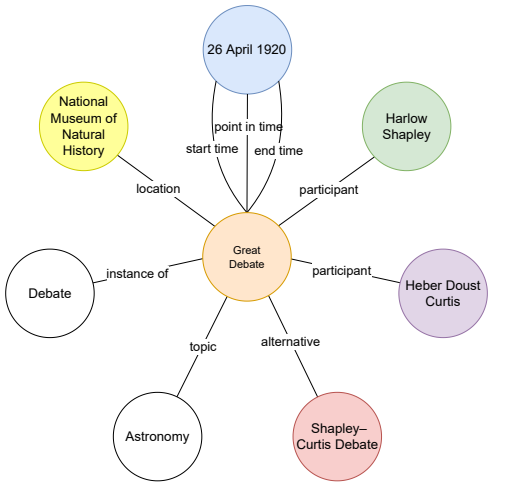

Figure 1: Given a graph, KG-to-text generation aims to describe the entities, relations, and its inherent structure via natural language text (grey callout). Corresponding graph-text components are color-coded.

Recently, pre-trained language models (LMs) have produced state-of-the-art results on the KG-to-text generation task (Ribeiro et al., 2020a; Chen et al., 2020). These models tend to first linearize a graph into a sequence of tokens, and fine-tune on pre-trained LMs such as BART (Lewis et al., 2020), GPT (Radford et al., 2019), or T5 (Raffel et al., 2020), treating the task similarly to a text-to-text task. Because of the performance gains caused by the self-supervised pre-training tasks, current work on KG-to-text has focused on developing pre-trained tasks and large-scale unlabeled graph-text

*These authors contributed equally.

corpora, replicating the success in the text-to-text domain (Chen et al., 2020; Ke et al., 2021). However, these works particularly focus on leveraging large amounts of pre-trained data for graph-to-text specific pre-trained tasks, e.g., recovering a masked text sequence based on a given complete KG.

Although recent work in KG-to-text has begun to combine LMs with a graph-aware approach (Ke et al., 2021), they do not adequately perform a graph-aware encoding structure, overlooking the KG's topological information.

We argue and show empirically that **without additional pre-trained tasks**, a fully graph-aware encoding combined with the coverage of pre-trained LMs such as BART (Lewis et al., 2020), can compete with and in some cases outperform those approaches which rely on additional pre-training. By doing so, we unload the burden of requiring vast amounts of data and computational resources required for pre-training.

We propose *GAP*, a KG-to-text framework which fuses **g**raph-**a**ware elements into existing **p**re-trained LMs, capturing the advantages brought forth by both model types. Our framework has two main components: (i) **Global Attention:** A graph's components are first encoded using an LM to capture their global semantic information, allowing the model to utilize the lexical coverage of pre-trained LMs (Davison et al., 2019; Gururangan et al., 2020; Vulić et al., 2020). (ii) **Graph-aware Attention**: Next, we encode local graph-wise information through a graph transformer (Veličković et al., 2018) augmented with a type encoder. Our framework attend to and update entities, nodes, or both. By proposing such a framework, where graph-aware components can be interchanged, we can begin exploring explainable generative models for the KG-to-text task.

We evaluate our proposed framework on two publicly available KG-to-text datasets: WebNLG (Gardent et al., 2017) and EventNarrative (Colas et al., 2021), achieving state-of-the-art results on various NLG metrics and demonstrating the value of our fully graph-aware LM based approach. Our contributions are as follows:

1. We propose a novel graph-aware framework for KG-to-text by introducing neighborhood-masked attention and connection type encoding into pre-trained LMs, capturing both local structural and global contextual information.

2. We begin to interpret KG-to-text generative models by drawing upon our framework and interchanging the various masking and type schemes, evaluating the output based on the variable graph topology.

3. We demonstrate on two datasets that by simply finetuning our models, which infuse graph-aware elements into existing LMs, one can even marginally outperform current state-of-the-art models which rely on several computationally expensive pre-training tasks.

We make our code publically available to motivate future research.

2 Related Work

2.1 KG-to-Text with Graph Transformers

Graph Neural Networks (GNNs) (Veličković et al., 2018) have shown to be effective at encoding graph data. For the KG-to-text task, recent works have leveraged GNNs to encode a graph's neighborhood information (Koncel-Kedziorski et al., 2019; Marcheggiani and Perez-Beltrachini, 2018; Ribeiro et al., 2020b; Schmitt et al., 2021; Guo et al., 2019; Jin et al., 2020) before decoding its corresponding textual representation. Other work instead choose a more global approach and base their encoder on a Transformer-based architecture (Vaswani et al., 2017), calculating self-attention from all the nodes in a graph (Zhu et al., 2019; Cai and Lam, 2020; Ke et al., 2021). Like previous work, we encode neighborhood information in the Graph-aware Attention module. Recently, graph convolution-based adaptors have been explored for Abstract Meaning Representation-to-text (Ribeiro et al., 2021). Unlike previous work, GAP is a framework for KG-to-text, where the KG's topology and masking scheme are not set. We encode local information through a GNN, while harnessing pre-trained LMs for a node's global information. While there has been work examining the effect of encoding a node's relative position (Shaw et al., 2018; Schmitt et al., 2021), we instead encode *type*, arguing that a KG's textual description is weighted based on its different types of connections, and empirically show its effect on KG-to-text generation.

2.2 KG-to-Text with Pre-trained LM

With the advent of pre-trained LMs such as BART (Lewis et al., 2020), T5 (Raffel et al., 2020), and GPT (Radford et al., 2019), these models have

been directly adapted and fine-tuned for the KG-to-text task and in some cases outperformed GNN-based models (Ribeiro et al., 2020a; Kale and Rastogi, 2020; Chen et al., 2020; Mager et al., 2020). While work has begun to explore combining such pre-trained models with transformer-based architectures which encode node information (Ke et al., 2021), they assume connectivity between all nodes and do not leverage updating relation information. Instead, here we propose a framework which combines pre-trained models with graph-aware encoders which are specifically neighborhood-based and dependent on a given graph's topology.

3 Problem Statement

We aim to generate texts that describe a given KG. We define a KG to be a multi-relational graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of entity vertices and $\mathcal{E} \subset \mathcal{V} \times \mathcal{R} \times \mathcal{V}$ is the set of edges that connect entities with a relation from $\mathcal{R}$.

4 Proposed Framework

As our model is built on top of LMs such as BART, we first linearize the knowledge graph into a text string (Distiawan et al., 2018; Moryossef et al., 2019; Su et al., 2021). The linearization is a sequence of all triples in the KG, interleaved with tokens that separate each triple and the triple's components (head, relation, and tail). Figure 2 shows an example linearization for a small knowledge graph, along with its labeled components.

4.1 Global Attention

We then use a transformer encoder to contextualize the vector representations. The first module in each transformer layer is a self-attention over the linearized graph, which acts as a *Global Attention* and captures the semantic relationships between all tokens. The Global Attention can be initialized with a pre-trained LM. At the l-th layer, the self-attention is formulated as:

$$X_l = \text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{1}$$

Query, key, and value are computed via $Q = X_{l-1}W_l^Q, K = X_{l-1}W_l^K$, and $V = X_{l-1}W_{l-1}^V$. Here, $X_{l-1} \in \mathbb{R}^{n\times d}$ denotes the collection of vectors corresponding to the graph's tokens. The model's parameters are denoted by matrices W of size $d_k \times d_k$, where d_k is the dimension of word vectors.

4.2 Graph Aware Attention

While the *Global Attention* assumes connectivity between all graph components, KG adjacencies are sparse in nature. To capture this, we propose a *Graph-aware Attention* module, by first retrieving entity/relation vectors from the word vectors. Some entities or relations contain several words or repeat several times in the linearized graph. To get a single vector for each entity/relation, we add a pooling layer, which takes the average of the corresponding word vectors for each entity/relation. Hence, we get the graph representation matrix $X_l^g \in \mathbb{R}^{m\times d}$:

$$X_l^g = \text{pooling}(X_l) \tag{2}$$

Note, $m < n$, where m and n denote the number of graph components and number of tokens, respectively. In practice and for parallelization m will be a fixed number larger than this sum for all graphs in the dataset, and the graph representation can be accessed via masking. We propose a novel graph-aware attention on the graph representation X_l^g by introducing a neighborhood-based masking scheme and novel type encoder:

$$\tilde{X}_l^g = \text{Attn}_{M,T}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}} + M + \gamma(T)\right)V. \tag{3}$$

Here Q, K, V are constructed from X_l^g by multiplying it with their corresponding learnable parameter W. While $M \in \mathbb{R}^{m\times m}$ is a mask that encodes the desired graph structure, and $\gamma(T) \in \mathbb{R}^{m\times m}$ is the type encoding matrix. Note, each row of Q, K, and V correspond to an element from the graph (an entity or a relation), and before applying a softmax in each row of $QK^\top$, we can mask/modify the scores based on the graph topology. For instance, $M_{ij} = -\infty$ forces the item i to not attend to item j or the value at $\gamma(T)_{ij}$ can add a bias to the attention score based on the type of connection between items i and j. We exploit this capacity to inject graph-awareness by adding a masking matrix M and type encoding matrix $\gamma(T)$.

4.2.1 Graph Topology Encoding

The proposed matrix $M \in \mathbb{R}^{m\times m}$ encodes the graph topology by assigning $-\infty$ where attention is blocked and 0 otherwise. M can be thought of as a generalized adjacency matrix for graph $\mathcal{G}$, which has both nodes and edges as its rows and columns. Hence, to encode neighborhood information for

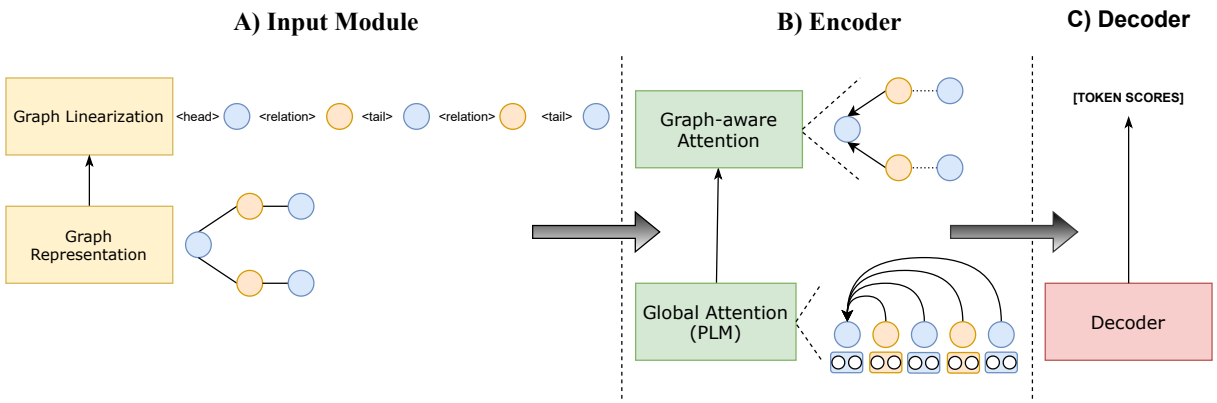

Figure 2: Overview of the Graph-aware framework for graph-to-text generation. Given a KG, we first transform the graph into its appropriate representation before linearizing the graph. Next, each node of the KG is encoded via a global attention, followed by a graph-aware attention, ultimately being decoded into a sequence of tokens.

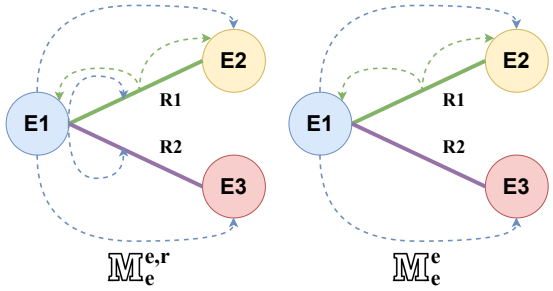

Figure 3: **Left:** $M_e^{e,r}$ mask, where E1 attends to its neighboring entities and relations, while R1 only attends to its neighboring entities. **Right:** M_e^{e} mask, where E1 and R1 only attend to their neighboring entities.

an entity, we can modify its corresponding row in M to have the value 0 for its neighbors, and $-\infty$ otherwise. As the rows and columns of M contain relations, we also have the capacity to let relations attend to their neighboring entity or relations.

From a graph topology perspective, we have several design choices for the matrix M. We can let entities attend to neighboring entities, neighboring relations, or both. We also have these same options for when relations are playing the query role; that is, when choosing which components should relations attend to. For ease of reference and discussion, superscript denotes neighborhood types for entities, while subscript for relations, e.g. $M_{e,r}^{e,r}$. For instance, when entities attend to neighboring entities and relations, but relations only attend to entities, we denote the masking matrix by $M_e^{e,r}$. Figure 3 illustrates two such matrices via a graph and its attending components.

4.2.2 Connection Type Encoding

In contrast to M which encodes the general graph topology, we also introduce a new type encoding $T \in \mathbb{R}^{m \times m}$, designed for biasing the attention values between the different graph components based on their connection type. For instance, when an entity e is attending to its neighbor entities $\{e_i\}$ and relations $\{r_i\}$, we encode the two connection types and bias their attention scores. Type information is stored in a matrix T, and we then use an embedding lookup $\gamma : \mathbb{Z} \rightarrow \mathbb{R}$ to learn scalar embeddings for the types in T.

We define type T_{ij} between query i and key j based on two factors: (i) whether the two items are connected and (ii) the type of each item, i.e. whether the connection is entity–entity, entity–relation, relation–entity, or relation–relation:

$$T_{ij} = \begin{cases} 0 & \text{if there's no connection,} \\ 1 & \text{if } i \text{ and } j \text{ are neighboring entities,} \\ 2 & \text{if } \{i, j\} \text{ is an \{entity,edge\} pair,} \\ 3 & \text{if } i \text{ and } j \text{ are adjacent relations.} \end{cases} \tag{4}$$

The model then has the capacity to modify its attention scores based on the graph's connection types. Intuitively, this capacity would allow us to interpolate between different choices of M, or in the extreme case it can push model $M_{e,r}^{e,r}$, to simulate any of the other more restrictive masks. For ease of reference, we explicitly state the type encoding whenever used.

Finally, after producing the new graph representation $\tilde{X}_l^g$ with equation (3), we *gather* the word representations from the graph representation, adding the new representations as a residual

Dataset	#KG-text pairs (Train/Valid/Test)
WebNLG	34,352 / 4,316 / 4,224
EventNarrative	179,543 / 1,000 / 22,441

Table 1: Statistics of the supervised KG-to-Text datasets used for experimenting.

to X_l, and generate the output from the l-th layer:

$$\tilde{X}_l = \text{gather}(\tilde{X}_l^g) + X_l \quad (5)$$

5 Experiments

5.1 Datasets

We evaluate our framework on two KG-to-text supervised datasets, namely: WebNLG (Gardent et al., 2017) and EventNarrative (Colas et al., 2021). We experiment with different configurations on the graph representation, attention mask, and type encoding on the WebNLG dataset, taking the best performing models to experiment further on EventNarrative. This is because of computational constraints caused by the size of EventNarrative.Table 1 outlines the statistical differences between the two datasets. We use the official data split for both.

WebNLG is a crowd-sourced RDF triple-to-text dataset manually crafted by human annotators. The dataset contains graphs from DBpedia (Auer et al., 2007) with up to 7 triples paired with one or more reference texts. Categories include airport, astronaut, building, city, food, artist, and politician. As in Chen et al. (2020) and Ke et al. (2021), we use the 2.0 release from GitLab [1] and replace underscores with spaces and split camel case entity and relation names into multiple tokens.

EventNarrative is an automatically generated large-scale event-centric KG-to-text supervised dataset. Event KGs are extracted from Wikidata (Vrandečić and Krötzsch, 2014) and EventKG (Gottschalk and Demidova, 2018), which are then matched to Wikipedia sentences. EventNarrative contains a larger number of unique KG components compared to WebNLG. We tokenize the KG-text pairs as in Ribeiro et al. (2020a) and Colas et al. (2021), adding special tokens to the vocabulary delineating the subject <*S*>, predicate <*P*>, and object <*O*> components of the KG.

5.2 Implementation and training details

We chose to use BART as our pre-trained LM (Lewis et al., 2020), and initialize its respective parameters with the Hugging Face's pre-trained bart-base checkpoint [2]. We left the default hyperparameters on the Global Attention module (BART) due to limited computational resources, instead experimenting on the Graph-aware attention module. As followed by Ke et al. (2021) and BART, we used a Byte-Pair Encoding (BPE) vocabulary (Radford et al., 2019) with a size of 50,265. The model's parameters were optimized via Adam (Kingma and Ba, 2015), with a batch size of 16, a learning rate of 3e-5, and a maximum graph size of 50 and 60 for WebNLG and EventNarrative, respectively. As in (Colas et al., 2021), we set the max output size to 512 for all experiments on EventNarrative because the corresponding labeled text can be up to 502 tokens. BLEU score on the validation set was used for model selection. Each model was trained on two NVIDIA RTX 2080 Ti GPUs.

When evaluating, we follow the existing work for KG-to-text and report the model's performance with BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), and ROUGE-L (Lin, 2004) scores as the automatic natural language generation (NLG) metrics.

5.3 Baselines

Fine-tuned LM. To evaluate the effect of the graph-aware attention module in our framework, we compare with a vanilla fine-tuned BART LM, which is not additionally pre-trained on any graph-text specific task. We do so for both WebNLG and EventNarrative, noting that for EventNarrative such a baseline is the state-of-the-art.

Pre-trained KG-to-Text Models. We further compare our framework with models which have pre-trained LMs on additional tasks, including KGPT (Chen et al., 2020) and JointGT (Ke et al., 2021). KGPT performs an additional KG-to-text generation pre-training task on KGText, a loosely-supervised large-scale KG-to-text dataset, before finetuning. JointGT performs three additional pre-training tasks for KG reconstruction, text reconstruction, and KG-text alignment on the KGText dataset before finetuning. For a fair comparison with JointGT, we also compare our results to JointGT's BART pre-trained task, where they perform an additional text infilling and sentence permutation task on KGText.

[1] https://gitlab.com/shimorina/webnlg-dataset

[2] https://huggingface.co/facebook/bart-base

Model	Pre	#Param	BLEU	METEOR	ROUGE
GCN (Marcheggiani and Perez-Beltrachini, 2018)	No	-	60.80‡	42.76 ‡	71.13‡
Shimorina and Gardent (2018)	No	-	61.00♯	42.00♯	71.00♯
KGPT w/o pretrain	No	177M	62.30 ‡	44.33 ‡	73.00‡
KGPT	Yes	177M	64.11‡	46.3‡	74.57‡
BART	Yes	140M	64.55	46.51	75.13
JointGT (BART) - w/ BARTPretrain	Yes	160M	64.60†	46.78†	75.74†
JointGT (BART) - w/ JointGTPretrain	Yes	160M	65.92†	**47.15**†	76.1†
GAP (**Ours**) - $M_e^{e,r}$	No	153M	65.92	46.81	76.22
GAP (**Ours**) - $M^{e,r} + \gamma$	No	153M	**66.20**	46.77	**76.36**

Table 2: Performance comparison on WebNLG. KGPT and JointGT, marked with † and ‡, re-printed from Chen et al. (2020) and Ke et al. (2021), have been pre-trained on one and three additional tasks. We report and mark results from Shimorina and Gardent (2018) with ♯. We report our best models with and without type encoding for comparison, which have approximately the same number of parameters.

Model	BLEU	METEOR	ROUGE	BERTScore
BART	31.38	26.68	62.65	93.12
T5	12.8	22.77	52.06	89.59
JointGT	31.19	26.58	**64.91**	**93.68**
$M_e^{e,r}$	34.02	26.93	62.90	93.13
$M^{e,r} + \gamma$	**35.08**	**27.50**	64.28	93.38

Table 3: Performance comparison on EventNarrative. We compare to the pretrained baselines, T5 and BART, reprinted from (Colas et al., 2021), and adapt JointGT (Ke et al., 2021) to the dataset.

GAP	BLEU	METEOR	ROUGE
$M_e^{e,r}$	**65.92**	46.81	76.22
$M^{e,r}$	65.86	**46.86**	**76.28**
M_e^{e}	65.11	46.33	75.62
$M_{e,r}^{e,r}$	64.64	46.17	75.04

Table 4: Experimental results of the different masks applied to the WebNLG test set.

5.4 Main results

Table 2 and Table 3 show our results on the WebNLG and EventNarrative datasets, respectively. On both datasets, we observe improvements over existing LM-based models with GAP. For BLEU score on WebNLG, we observe a +5.20% improvement over the state-of-the-art without any pre-training (Shimorina and Gardent, 2018) and a +1.65% improvement over BART. This improvement suggests that the graph-aware component of GAP makes use of the local neighborhood information when encoding graph components.

We outperform both KGPT and JointGT (on WebNLG), which rely on additional pre-training tasks for graph-text reconstruction and alignment. On BLEU score, we observe an improvement of +1.81% and 2.09% over KGPT, and +1.32% and 1.6% over JointGT (with BARTPretrain). Further, our $M^{e,r}$ with *Type Encoding* model outperforms JointGT (with JointGTPretrain) by 0.28% without the need for any additional pre-training. JointGTPretrain refers to all three pre-trained tasks described in Ke et al. (2021). Instead of pre-training, we fill the gap with a modification to the encoder structure such that the model adapts to the graph structure. To summarize, we have shown that when adapting pre-trained language models such as BART, a careful modification of the encoder structure can better align the LM with the new task.

On EventNarrative, for model $M_e^{e,r}$ we achieve an improvement of +3.70%, +0.82%, +1.63% on BLEU, METEOR, and ROUGE, relative to BART, further demonstrating that the graph-aware structure and type encoder can perform comparatively well on large and more complex graphs. We note a similar trend to WebNLG, where the type encoder can give an additional performance improvement to the graph-structure component of the model. For comparison, we adapt JointGT to EventNarrative, using the hyperparameters from Ke et al. (2021). We note all models have similar BERTScores.

6 Analysis

6.1 Ablation Studies

We explore different maskings and type encodings for the graph-aware attention module on WebNLG, summarized on Table 4 and Table 5.

Masking Scheme. The first design choice in our graph-aware attention module is the mask-

GAP w/ γ	BLEU	METEOR	ROUGE
$M_e^{e,r}$	65.34	46.31	75.59
$M^{e,r}$	**66.20**	**46.77**	**76.36**
M_e^e	65.24	46.49	75.44
$M_{e,r}^{e,r}$	65.43	46.54	75.75

Table 5: The results of different variations of our model with type encoding on the WebNLG test set.

GAP	#Triples	
	1-3	4-7
$M_e^{e,r}$	71.48	61.53
$M^{e,r}$	71.28	61.59
M_e^e	70.18	61.05
$M_{e,r}^{e,r}$	69.74	60.57

Table 6: BLEU scores for the different masks applied to the WebNLG test set for different graph sizes.

ing scheme. From bottom to top on Table 4, our first observation is that when relations directly attend to the neighboring relations, the performance drops by 1.28%, the largest difference. In fact, the results significantly improve when we completely block attention on relations (M_e^e). However, for the entities, it is always best to attend to their edges (relations) as well as their neighboring entities. The top two results are comparable (0.06% difference in BLEU score), and each one could be considered the best performing model depending on the evaluation metric. For relations, it might be somewhat helpful to not attend to neighboring relations, while for entities, attending to the relations will lead to better results (+0.81%).

Type Encoder. Table 5 shows the effect of type encoding on the results on WebNLG. To better understand the effect of type encoding on each of the models, we compare Table 4 with Table 5. Recall that the type encoding $\gamma(T)$ for each model depends on the connections that exists in the model graph structure. For instance, the most general model $M_{e,r}^{e,r}$ has all four possible connection types encoded by equation (4), while the model with $M = M^{e,r}$ only has two types, which can be encoded by a restriction of equation (4). According to Table 4, the model $M_{e,r}^{e,r}$ performs worst without type encoding. However, because of its generality, i.e. having all the possible connection types, it is possible for this model to drift toward better configurations with the help of $\gamma(T)$. The results in Table 5 help support these insights for model $M_{e,r}^{e,r}$. Type encoding allows this model to simulate what we observed is best in the previous section, i.e. relations are better off not to attend to relations, whereas entities can attend to both while paying less attention to relations. This nuanced behavior seems to be achievable only via type encoding. Results for model with $M = M^{e,r}$ and type encoding also point towards this; type encoding seems to facilitate a non-uniform attention distribution based on the type and produces a better result.

6.2 KG Size

As in Ke et al. (2021), we divide the WebNLG test set into two subsets (1-3 and 4-7 triples) to compare the performance of our different masking configurations. Table 6 shows that while all configurations perform similarly for smaller larger graphs, the difference in performance is clearer on smaller graphs, where $M_e^{e,r}$ performs +1.74% better than $M_{e,r}^{e,r}$, suggesting that relations paying attention to relations can add too much complexity to the model, especially on simpler graph structures.

6.3 Interpretability

We begin to interpret KG-to-text models by analyzing the graph-attention weights induced by each graph structure on a per-sample basis, analogous to analyzing node-to-node attention weights in the KG question-answering domain (Yasunaga et al., 2021). By introducing a framework to the KG-to-text task, we can condition the changes in the output text on the different components of the framework, including the masking and type encoder. We can then observe the differences in the output text based on the graph's topological structure or what relations and entities attend to.

In Figure 4 we show an example KG representing *Aenir*, an Australian fantasy novel, with its relations (orange) and entities (blue) along with the attention heatmaps and outputs from two of our framework decisions. The left (a) heatmap and output corresponds to our best performing model without type encoding, $M_e^{e,r}$, while the right (b) corresponds to M_e^e. We choose these two masking configurations, because the attention-weight differences are apparent.

From (a), entities attend to both entities and relations, whereas relations only attend to entities. Interestingly, the attention distribution appears uniform across all graph components (both for entities and relations). From (b) we see a similar uniform distribution across entities and relation attending

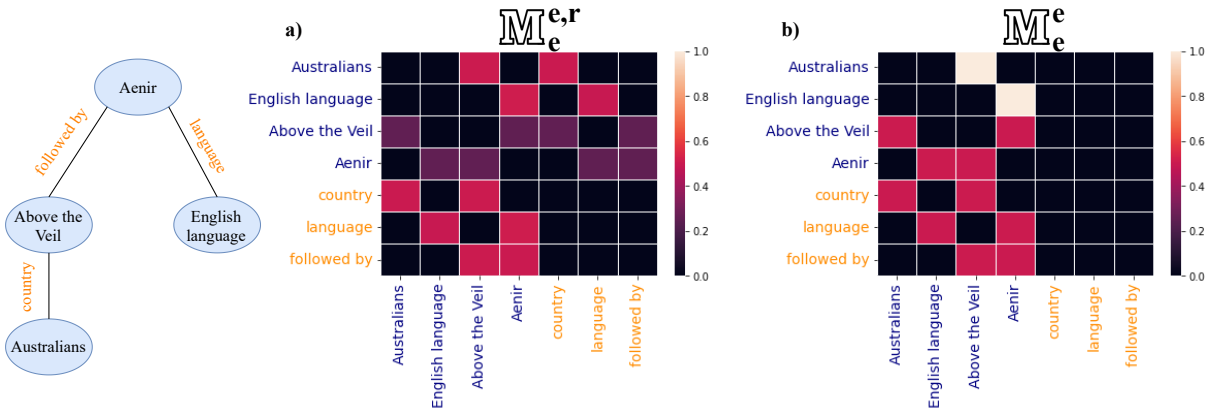

Figure 4: Interpreting KG-to-text models via analyzing graph attention weights, which the graph-aware encoder activates. We show each model's output for further emphasis.

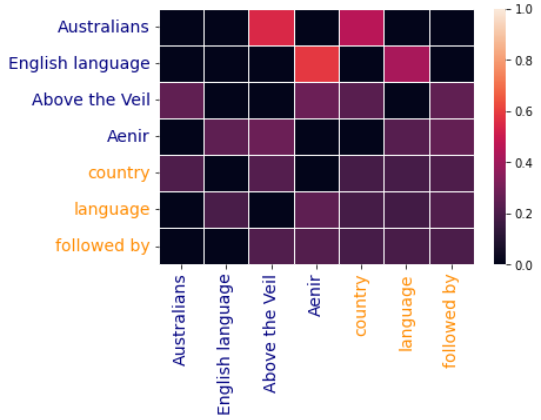

Figure 5: An additional case study of the graph attention weights for model $M_{e,r}^{e,r}$ with type encoding.

to only entities. Thus, in (a), while relation 'country' attends to 'Australians' and vice-versa, in (b) 'Australians' does not attend to 'country', perhaps giving a difference in the output, as the final output in (a) contains 'from the country of Australia' while the text in (b) does not. Moreover, in both (a) and (b) 'Above the Veil' is the subject of the second clause. However, 'Above the Veil' attends to 'country' only in (a), therefore influencing (a)'s output of '*Above the Veil* which is from the *country* of Australia'. Instead, (b) introduces some redundancy in its second clause instead of transcribing new information from the KG.

Figure 5 shows the output sentence, and the attention heatmap produced by our most general model with $M = M_{e,r}^{e,r}$ and type encoding, on the graph shown in Figure 4. We examine the differences between this model, referred to as model (1), and the model with $M = M_e^{e,r}$ and no type encoding, referred to as model (2). First, note that in terms of BLEU score (1) performs slightly worse than (2), however a human annotator may rank (1) over (2), as (1) is more concise while communicating the same information. For example, (1) uses the word 'sequel to' rather than 'followed by' and 'published in' instead of 'from the country', which can sound more natural to humans. Particularly, *Australians* pays less attention to *country*, compared to model (2), perhaps hinting at this result. Our framework provides a first step in interpreting this result by allowing one to compare different attention-weights across multiple models. With this in mind, we call upon future work to design more specific evaluation metrics for the KG-to-text task.

7 Conclusion

We presented GAP, a graph-aware language model framework for KG-to-text generation. Our framework instills the local information captured by graph attention into the global contextualized word vector representation within pre-trained LMs. We demonstrated multiple configurations of our framework by introducing a *graph-aware attention masking scheme* and novel *type encoder* module, and through qualitative analysis showed that GAP outperforms existing KG-to-text models, including those that rely on additional auxiliary pre-training tasks. By closely examining the different framework configurations, we introduce the capacity to interpret KG-to-text outputs through a graph's attention structure and topology.

8 Broader Impacts

GAP provides researchers with a state-of-the-art framework for KG-to-text models. Though we experiment with supervised baselines which include a handcrafted dataset, WebNLG, and an automatically generated dataset, EventNarrative, repositories of structured data exist in the clinical (Johnson et al., 2016), medical (Bodenreider, 2004), and news crises (Leetaru and Schrodt, 2013; Ward et al., 2013) domains. By transforming clinical data into natural language narratives, patients with low health-literacy can benefit by more easily understanding their electronic medical records (EMRs), and doctors can more easily transcribe patient data for future use cases, i.e. connecting such data to the medical literature. Such models can also help analysts more easily understand crises data from various news sources, in turn helping them evaluate cause-effect relationships and detect misinformation. While malicious actors can exploit generative models for disinformation, we discourage the use of GAP in generating such data and openly release our model to help combat such efforts.

References

Oshin Agarwal, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. 2021. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3554–3565, Online. Association for Computational Linguistics.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Rajarshi Bhowmik and Gerard de Melo. 2018. Generating fine-grained open vocabulary entity type descriptions. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 877–888.

Olivier Bodenreider. 2004. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl_1):D267–D270.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

Deng Cai and Wai Lam. 2020. Graph transformer for graph-to-sequence learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7464–7471.

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.

Anthony Colas, Ali Sadeghian, Yue Wang, and Daisy Zhe Wang. 2021. Eventnarrative: A large-scale event-centric dataset for knowledge graph-to-text generation.

Joe Davison, Joshua Feldman, and Alexander M Rush. 2019. Commonsense knowledge mining from pretrained models. 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 1173–1178.

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

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.

Simon Gottschalk and Elena Demidova. 2018. Eventkg: A multilingual event-centric temporal knowledge graph. In *European Semantic Web Conference*, pages 272–287. Springer.

Jian Guan, Yansen Wang, and Minlie Huang. 2019. Story ending generation with incremental encoding and commonsense knowledge. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6473–6480.

Zhijiang Guo, Yan Zhang, Zhiyang Teng, and Wei Lu. 2019. Densely connected graph convolutional networks for graph-to-sequence learning. *Transactions of the Association for Computational Linguistics*, 7:297–312.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don't stop pretraining:

Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360.

Shizhu He, Cao Liu, Kang Liu, and Jun Zhao. 2017. Generating natural answers by incorporating copying and retrieving mechanisms in sequence-to-sequence learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 199–208.

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

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

Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-Wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9.

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

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.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

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.

Kalev Leetaru and Philip A Schrodt. 2013. Gdelt: Global data on events, location, and tone, 1979–2012. In *ISA annual convention*, volume 2, pages 1–49. Citeseer.

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

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

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

Diego Marcheggiani and Laura Perez-Beltrachini. 2018. Deep graph convolutional encoders for structured data to text generation. In *INLG*.

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, Volume 1 (Long and Short Papers)*, pages 2267–2277.

Vaishali Pal, Manish Shrivastava, and Irshad Bhat. 2019. Answering naturally: Factoid to full length answer generation. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 1–9.

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.

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

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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. *Journal of Machine Learning Research*, 21(140):1–67.

Leonardo F. R. Ribeiro, Yue Zhang, and Iryna Gurevych. 2021. 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.

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

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

Martin Schmitt, Leonardo FR 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.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468.

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

Yixuan Su, David Vandyke, Sihui Wang, Yimai Fang, and Nigel Collier. 2021. Plan-then-generate: Controlled data-to-text generation via planning. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 895–909.

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

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Ivan Vulić, Edoardo Maria Ponti, Robert Litschko, Goran Glavaš, and Anna Korhonen. 2020. Probing pretrained language models for lexical semantics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7222–7240.

Michael D Ward, Andreas Beger, Josh Cutler, Matthew Dickenson, Cassy Dorff, and Ben Radford. 2013. Comparing gdelt and icews event data. *Analysis*, 21(1):267–297.

TH Wen, M Gašić, N Mrkšić, LM Rojas-Barahona, PH Su, D Vandyke, and S Young. 2016. Multi-domain neural network language generation for spoken dialogue systems. In *2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2016-Proceedings of the Conference*, pages 120–129.

David S Wishart, Yannick D Feunang, An C Guo, Elvis J Lo, Ana Marcu, Jason R Grant, Tanvir Sajed, Daniel Johnson, Carin Li, Zinat Sayeeda, et al. 2018. Drugbank 5.0: a major update to the drugbank database for 2018. *Nucleic acids research*, 46(D1):D1074–D1082.

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.

Alexander Yates, Michele Banko, Matthew Broadhead, Michael J Cafarella, Oren Etzioni, and Stephen Soderland. 2007. Textrunner: open information extraction on the web. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pages 25–26.

Hao Zhou, Tom Young, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2018. Commonsense knowledge aware conversation generation with graph attention. In *IJCAI*, pages 4623–4629.

Jie Zhu, Junhui Li, Muhua Zhu, Longhua Qian, Min Zhang, and Guodong Zhou. 2019. Modeling graph structure in transformer for better amr-to-text generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5459–5468.

A Hyperparameter Details

Table 7 provides the model hyperparameter settings used for experimenting on both the WebNLG and EventNarrative datasets. We keep all listed hyperparameters constant with respect to the GAP configurations. We increase *num nodes* for the EventNarrative dataset due to the properties of the dataset, i.e. the possibility of having graphs composed of more than seven triples. We also set the *eval period* to 5,000 for EventNarrative due to its size, containing approximately 22,000 samples in its test set.

Hyperparameter	**WebNLG**	**EventNarrative**
Learning Rate	2.00E-05	2.00E-05
Warmup Steps	1600	1600
Eval Period	500	**5000**
Beam Size	5	5
Length Penalty	1	1
Optimizer	Adam	Adam
ϵ	1.00E-08	1.00E-08
Num Nodes	50	**60**
Num Relations	60	60
Embedding Size	128	128
Num Global Layers	6	6
Num Graph-aware Layers	6	6
Batch Size	16	16

Table 7: Hyperparameters for GAP on both the WebNLG and EventNarrative datasets.

B Additional Experimental Results

We provide additional experimental results on both WebNLG and EventNarrative for the proposed GAP framework for reference and further analysis.

B.1 Graph Length

Here we examine a comparative study to that of Table 6 for the EventNarrative dataset. Table 8 reveals an exponential decay in BLEU score, with lengths 1-3, 4-7, and 7+ having 44.48%, 23.86%, 11.47%, respectively. Compared to WebNLG, the BLEU scores are significantly lower, suggesting that EventNarrative is a more challenging dataset. Table 9 gives a brief synopsis of the dataset sizes with respect to the number of triples. Compared to WebNLG which has no KGs greater than length 7, EventKG contains over 1,000 KGs larger than length 7, making the dataset more diverse.

GAP	#Triples		
	1-3	4-7	7+
$M_e^{e,r}$	44.48	23.86	11.47

Table 8: BLEU scores for the EventNarrative test set for different graph sizes.

Datasets	#Triples		
	1-3	4-7	7+
WebNLG	1,017	583	0
EventNarrative	16,103	5,152	1,184

Table 9: Distribution for number of triples in both the WebNLG and EventNarrative datasets.

B.2 Entity Accuracy

To give more insight into KG-to-text generation with GAP, we provide the results for *entity accuracy*. We define *entity accuracy* to be the number of entities from the KG that appear in the generated text over those that appear in the reference text. Table 10 shows that all models perform exceedingly well in generating the correct entities from their respective KGs, suggesting that future KG-to-text research should focus on sentence structure and descriptors, i.e. quantifiers and determiners.

Datasets	Accuracy	
	w/o $\gamma(T)$	w/ $\gamma(T)$
$M_e^{e,r}$	94.06	94.04
$M^{e,r}$	93.99	94.48
M_e^{e}	93.64	94.50
$M_{e,r}^{e,r}$	93.82	94.28

Table 10: Entity accuracy on the WebNLG test set.

C Additional Examples and Error Analysis

We now present example outputs generated by GAP both on the WebNLG and EventNarrative dataset in Tables 11 and 12 below.

C.1 WebNLG

We showcase five different examples from the WebNLG test set output by our $M^{e,r} + \gamma(T)$ (Prediction 1) and $M_{e,r}^{e,r} + \gamma(T)$ (Prediction 2) models. As can be seen in all the examples, GAP is able to generate fluent and complete sentences. In the first two examples, the output from both models are identical. The outputs from the third example can be viewed as paraphrases of one another,

where Prediction 1 mentions *'US national'* while Prediction 2 instead uses the adjective *'American'* to convey the same information. Furthermore, in both predictions we learn that *'Alan Bean'* was a *'test pilot'* and *'selected by NASA'* but in slightly different formats. In the fourth example, Prediction 2 is missing the name of the rock band, *'NRBQ'*, while maintaining the rest of the information. Like the third example, the predictions in the fifth example are paraphrases.

C.2 EventNarrative

Because of the length of output in EventNarrative, we present four different types of examples to elaborate on the limitations of KG-to-text models. Here, we show example outputs from our $M_e^{e,r}$ (Prediction 1) and $M^{e,r} + \gamma(T)$ (Prediction 2) models. In the first example, we observe a **contradiction** in both Prediction 1 and 2: the gubernatorial candidate was a democratic nominee, while our predictions conveyed otherwise. The second example shows two predictions which are identical, both **missing information**, specifically *'ozone park'* and *'for three - year - olds and up'*. Upon further inspection, these two pieces of information are not within the KG. Similarly, in the third example the only piece of information missing from the predictions, namely *'cork county board'*, is not part of the KG. This example also contains **invalid information**, *'112th'* instead of *'103rd'*. The last example also contains **invalid information** regarding the dates in both predictions. Additionally, Prediction 2 is **missing information** about the *'village of ignacewo'*.

Prediction 1	Amsterdam Airport Schiphol serves the city of Amsterdam and is -3.3528 above sea level . The runway name is 18L/36R Aalsmeerbaan and it has a length of 2014.0 .
Prediction 2	Amsterdam Airport Schiphol serves the city of Amsterdam and is -3.3528 above sea level . The runway name is 18L/36R Aalsmeerbaan and it has a length of 2014.0 .
Reference	Amsterdam Airport Schiphol is -3.3528 above sea level , has a runway name 18L/36R'Aalsmeerbaan which is 2014.0 in length and serves the city of Amsterdam .
Prediction 1	Baked Alaska is from Hong Kong and the United States . The main ingredients are meringue , ice cream , sponge cake or Christmas pudding .
Prediction 2	Baked Alaska is from Hong Kong and the United States . The main ingredients are meringue , ice cream , sponge cake or Christmas pudding .
Reference	Baked Alaska comes from both Hong Kong and the United States . The main ingredients are Meringue , ice cream , sponge cake or Christmas pudding .
Prediction 1	Alan Bean is a US national who was born in Wheeler , Texas . He served as a test pilot before being selected by NASA in 1963 . He is now retired .
Prediction 2	Alan Bean is an American test pilot who was born in Wheeler , Texas . He was selected by NASA in 1963 . He is now retired .
Reference	The American test pilot Alan Bean (born in Wheeler , Texas) was selected by NASA in 1963 . He is now retired .
Prediction 1	Al Anderson is a member of rock band NRBQ . Rock music originated from country music which originated from blues music . A musical fusion of rock music is bhangra music .
Prediction 2	Al Anderson plays rock music which originated from blues and country music . Bhangra music is part of the fusion genre , partly coming from Rock music which uses the banjo .
Reference	Al Anderson plays with the rock band NRBQ . Rock has its origins in the blues and country music , where the banjo is played , and Bhangra is a rock fusion .
Prediction 1	Paulo Sousa is the manager of ACF Fiorentina . He has played for Inter Milan , Juventus FC and Maccabi Tel Aviv F.C .
Prediction 2	Paulo Sousa is the manager of ACF Fiorentina . He has played for Inter Milan , Maccabi Tel Aviv FC and Juventus FC .
Reference	Paulo Sousa once played for Inter Milan and Juventus FC . He now manages ACF Fiorentina and plays for Maccabi Tel Aviv F.C .

Table 11: Examples of output texts generated from $M^{e,r} + \gamma(T)$ (Prediction 1) and $M^{e,r}_{e,r} + \gamma(T)$ (Prediction 2) on the WebNLG test set.

Prediction 1	the 1971 mississippi gubernatorial election was held on 02 november 1971 . incumbent governor bill waller won re - election to a second term , defeating the democratic nominee , former speaker of the chamber of deputies thomas h . williams , by a wide margin .
Prediction 2	the 1971 mississippi gubernatorial election took place on 02 november 1971 , in order to elect the governor of mississipp . incumbent governor bill waller , who was first elected in 1972 , faced no opposition from democrats and avoided a runoff .
Reference	the 1971 mississippi gubernatorial election took place on 02 november 1971 for the post of governor of mississippi. democrat bill waller was elected into office and served as governor of mississippi from 1972 to 1976.
Prediction 1	the sport page handicap is an american thoroughbred horse race run annually at aqueduct racetrack in queens , new york .
Prediction 2	the sport page handicap is an american thoroughbred horse race run annually at aqueduct racetrack in queens , new york .
Reference	the sport page handicap was an american thoroughbred horse race run annually at aqueduct racetrack in ozone park, queens, new york for three - year - olds and up.
Prediction 1	the 1991 cork senior hurling championship was the 112th staging of the cork premier hurling competition , and the 86th staging by a team from cork .
Prediction 2	the 1991 cork senior hurling championship was the 112th staging of the cork premier club hurling competition since its establishment in 1887 .
Reference	the 1991 cork senior hurling championship was the 103rd staging of the cork senior hurling championship since its establishment by the cork county board in 1887.
Prediction 1	the first battle of ignacewo was one of the first battles of the january uprising . it took place on january 28 , 1863 , near the village of ignakewo , konin county in southwestern corner of russian - controlled congress poland .
Prediction 2	the first battle of ignacewo was one of the first battles of the january uprising . it took place on january 6 , 1863 , near the village of konin , in congress poland .
Reference	the first battle of ignacewo was one of many clashes of the january uprising. it took place on may 8, 1863, near the village of ignacewo, konin county, which at that time belonged to russian empire's congress poland.

Table 12: Examples of output texts generated from $M_e^{e,r}$ (Prediction 1) and $M^{e,r} + \gamma(T)$ (Prediction 2) on the EventNarrative test set.