Hallucination Mitigation in Natural Language Generation from Large-Scale Open-Domain Knowledge Graphs

Anonymous EMNLP submission

Abstract

In generating natural language descriptions for knowledge graph triples, prior works used either small-scale, human-annotated datasets or datasets with limited variability of graph shapes, e.g., mostly star graphs. Graph-to-text models trained and evaluated on such datasets are largely neither refined nor assessed for more realistic large-scale, open-domain settings. We introduce a new dataset, GraphNarrative, that seeks to fill this gap. Fine-tuning transformer-based pre-trained language models (PLMs) has achieved state-of-the-art performance among graph-to-text models. However, this method suffers from information hallucination—the generated text may contain fabricated facts that are not present in input graphs. We propose a novel approach that, given a (graph, sentence) pair in the dataset, trims the sentence to eliminate portions that are not present in the corresponding graph, by utilizing the sentence's dependency parse tree. Experiment results verify this approach using both GraphNarrative and existing datasets.

1 Introduction

The task of *graph-to-text generation* aims to automatically generate natural language descriptions of knowledge graphs. A knowledge graph $\mathcal{G}$ stores factual information as subject-predicate-object triples, where each triple (s, p, o) corresponds to an edge from the subject entity s to the object entity o. The graph-to-text generation task entails, given a subgraph $G \subset \mathcal{G}$, generate a token sequence $Y = (y_1, ..., y_n)$ to describe G. This task can be accomplished by constructing machine learning models (Clive et al., 2021; Castro Ferreira et al., 2019; Trisedya et al., 2018). The input to such a model is a graph itself—a small fragment of triples from a knowledge graph, incoming from the outcome of some upstream operation, such as search, query, and data mining. The output is a textual sequence that describes the fragment of triples.

Verbalizing triples from knowledge graphs is crucial in a variety of tasks and applications, including systems created for querying knowledge graphs (Liang et al., 2021; Jayaram et al., 2016) as well as systems backed by knowledge graphs for question-answering (Zhou and Small, 2019; Ma et al., 2018) and fact discovery (Xian et al., 2019; Zhang et al., 2018). In these places, knowledge graph fragments must be conveyed to users in various forms, such as query results and discovered facts. Though a tiny part of a whole knowledge graph, such graph fragments can be complex and thus challenging to comprehend. Instead, presenting them in natural language can help end users understand them better.

In graph-to-text generation, the preciseness and naturalness of the textual narration of graph fragments are important. Generating high-quality text can be particularly challenging for *large-scale* and *open-domain* knowledge graphs. Specifically, benchmark datasets in this line of research either are *hand-crafted* and *monotonous*, e.g., WebNLG (Gardent et al., 2017a), or only include *simple, special* formations in narrated input fragments, e.g., EventNarrative (Colas et al., 2021) and TEKGEN (Agarwal et al., 2021). Existing graph-to-text models, being trained and evaluated on these datasets, are largely neither validated nor refined for more realistic large-scale, open-domain settings. Section 2 presents this analysis in detail.

This paper introduces GraphNarrative, a new dataset that fills the aforementioned gap between graph-to-text models and real-world needs. GraphNarrative consists of around 8.7 million (input graph, output text) pairs. The text in each pair is a Wikipedia sentence, whereas the corresponding graph comprises Freebase (Bollacker et al., 2008) entities and relations described in the sentence. The large-scale of both Wikipedia and Freebase, the linguistic variability in Wikipedia, and the complexity of sentences and corresponding graph structures

make this dataset more aligned with real-world scenarios. For instance, GraphNarrative's 8.7 million input graphs are in 7,920 distinct topological shapes and 22% of the 8.7 million are star graphs, in contrast to 94% and 96% in EventNarrative and TEKGEN, respectively. Section 3 articulates the details of GraphNarrative's creation.

Given the demonstrated efficacy of fine-tuning pre-trained language models (PLMs) in producing state-of-the-art results on graph-to-text (more details in Section 4), we adopt the same approach. As pointed out in (Agarwal et al., 2021; Dušek et al., 2018), though, this approach may suffer from information *hallucination*, i.e., the output texts may contain fabricated facts that are not present in input graphs. For example, given a two-triple input graph ((`Neff Maiava`, *date of birth*, `01 May 1924`), (`Neff Maiava`, *date of death*, `21 April 2018`)), (Agarwal et al., 2021) reported their model generates "Neff Maiava (1 May 1924 - 21 April 2018) was an Albanian actor". Not only the input does not mention Maiava's profession or citizenship, but also in the real-world he was an American Samoan wrestler instead.

Very few have considered how to mitigate hallucination in graph-to-text generation, except for (Agarwal et al., 2021; Wang et al., 2021) which attempted to address hallucination by further fine-tuning PLMs on WebNLG after fine-tuning on noisier automatically-extracted datasets. However, these studies did not quantify the prevalence of hallucination in their models' outputs. Nor did they provide experiment results or other evidence to verify the approach. We are the first to quantitatively measure the prevalence of hallucination in graph-to-text. We also developed a novel approach to mitigating hallucination by aiming at the problem's root—mismatch between graph and text in training data. Given a (graph, text) pair in GraphNarrative, the approach trims the text, i.e., a Wikipedia sentence, by eliminating portions that are not represented in the graph. This process is accomplished by analyzing the shortest paths between graph entities within the sentence's dependency parse tree. Details about the approach, named *sentence trimming*, are in Section 6.3.5.

We conducted comprehensive automatic and human assessments of text descriptions generated by fine-tuned PLMs (BART and T5). The automatic evaluation results consistently demonstrate that models perform better with the use of the sentence trimming, across the GraphNarrative, TEKGEN, and WebNLG datasets. It leads to the increment of 12 and 7 points in the BLEU score (Papineni et al., 2002) for the first two respective datasets. A T5-large model, which underwent initial fine-tuning on GraphNarrative with sentence trimming, achieved the state-of-the-art results on the WebNLG benchmark. Furthermore, human evaluation results show that sentence trimming on average reduces 1.4 entity hallucinations and 1 relation hallucination per text description.

The contributions of this paper are as follows. The dataset, source code and models are released at `https://anonymous.4open.science/r/graphnarrator-40F6` and will be made publicly available.

- A new dataset, GraphNarrative, that fills the gap between existing datasets and large-scale real-world settings.
- The first to quantify hallucinations produced by graph-to-text models.
- A novel approach, sentence trimming, to hallucination mitigation.
- Comprehensive experiments and evaluations that show the quality and utility of GraphNarrative, as well as the effectiveness of sentence trimming with state-of-the-art results.

2 Limitations of Existing Datasets

This section analyzes the limitations of current datasets. Graph-to-text models trained on such data may not be as effective on *large-scale* and *open-domain* knowledge graphs in the real world. Furthermore, evaluating models using these datasets may not reveal their weaknesses when facing such real-world knowledge graphs.

First, most previous models were trained on small hand-crafted datasets that contain limited entity types and relations. For instance, WebNLG includes 2,730 distinct entities and 354 distinct relations. In contrast, real-world knowledge graphs can be much larger. For example, according to (Heist et al., 2020), WikiData (Vrandečić and Krötzsch, 2014) has 52,252,549 entities, 2,356,259 classes, 6,236 relations, and 732,420,508 triples. The hand-crafted approach cannot scale to these massive knowledge graph, as it is impossible to manually write training graph-text pairs for so many different entity types, relations, and topic domains.

Second, the text descriptions in the instances of hand-crafted datasets such as WebNLG tend to follow monotonous templates, likely because

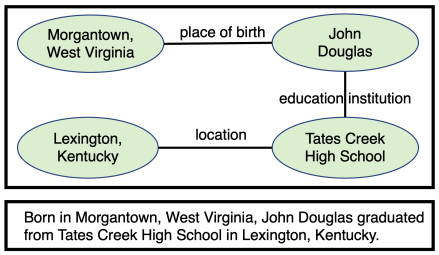

Figure 1: A (graph, sentence) pair in GraphNarrative

the examples were written by a small number of human contributors. This limits the capacity of trained models to use diverse expressions in narrating graph fragments. Such a lack of linguistic variability can hamper the overall usability of a text generation system.

Third, the graph fragments in existing datasets are largely limited to simple *star graphs* (each graph consisting of a center entity and some of its one-hop neighbors) or acyclic graphs (i.e., one or more trees). The graphs in WebNLG have 41 distinct topological shapes (Appendix C.4), out of which 34 are acyclic graphs. In automatically-generated datasets EventNarrative and TEKGEN, 94% and 96% of the graphs are star graphs, respectively. Another automatically-collected dataset, AGENDA (Koncel-Kedziorski et al., 2019), has only 2% star graphs. But it only contains 7 distinct relations in the special domain of scientific research. On the contrary, in practical scenarios the input fragments can be of *complex, general* rather than simple, special formations. While direct measurement is lacking, we used the graphs described in Wikipedia sentences as a proxy for gauging the shape diversity of graphs that need to be narrated. We manually analyzed the formations of graphs presented in 100 random Wikipedia sentences, and we found only 39 of the 100 graphs are star graphs. Similar but automatic analysis of the complete Wikipedia corpus (more details in Section 3, Figure 2) show that only 2 of the 10 most frequent graph formations [1] are star graphs.

3 The GraphNarrative Dataset

This section explains how we *automatically* generated our new dataset, GraphNarrative, for the graph-to-text task by aligning Wikipedia texts with Freebase. Note that the methodology could be applicable to other text corpora (instead of Wikipedia) and other knowledge graphs (instead of Freebase). This section also contrasts the dataset with existing benchmark datasets in order to demonstrate how it addresses current datasets' limitations.

[1] Or 3 out of the 10, depending on whether considering a 3-node path as a star or not.

3.1 Dataset Creation: Graph-Text Alignment

For each applicable Wikipedia sentence W, we create the corresponding subgraph G in Freebase, to form a graph-sentence pair (G, W) as one example instance in the dataset. One such example is in Figure 1. This is achieved by a process of detecting Freebase entities and edges from W, consisting of a *entity linking* (Shen et al., 2014) step followed by a *edge detection* step.

Entity linking: It maps a span of tokens in the Wikipedia sentence W to an entity e in Freebase. Our customized entity linking solution consists of coreference resolution (McCarthy and Lehnert, 1995), wikification (Csomai and Mihalcea, 2008), and Wikipedia-to-Freebase entity mapping. The entity mapping (more details in Section C.3) created 4,408,115 one-to-one mapping between English Wikipedia entities (i.e., articles) and Freebase entities, using a combination of three engineering methods—by using existing mapping in Freebase, by using Wikidata as the middle ground connecting Wikipedia and Freebase entities, and similarly by using DBpedia as the middle ground. For *wikification*, our simple approach maps a span of tokens in a Wikipedia article D to a Wikipedia entity, if the tokens exactly match either the entity's full title or any of the entity's wikilink anchor text in the same article D. For *coreference resolution*, we applied AllenNLP's (Gardner et al., 2017) coreference resolution (Lee et al., 2017) on Wikipedia articles to replace pronouns and aliases with corresponding entities. The results of aforementioned processes were put together—a Wikipedia entity appearance in a Wikipedia sentence, either originally as a wikilink or detected through wikification upon coreference resolution, leads to the detection of the corresponding Freebase entity via the mapping results.

Edge detection: Given the Freebase entities detected from a Wikipedia sentence W, it identifies Freebase edges between the entities such that the corresponding relations are described in W. Given a pair of such entities, if Freebase contains only one edge between them, our simple method assumes the corresponding relation is described in W. If Freebase has multiple edges between them, we include the edge whose label tokens overlap with W. If there are still multiple such edges, we include the

Dataset	Knowlege Graph	Text Corpus	Domain	Instances	Entities	Triples	Relation	Star Graphs
WebNLG	DBpedia	Handcraft	15 DBpedia categories	25,298	2,730	3,221	354	57%
AGENDA	N/A	Scientific abstract	Scientific research	40,720	159,691	177,568	7	**2%**
EventNarrative	Wikidata	Wikipedia	Events	224,428	305,685	649,337	672	94%
TEKGEN	Wikidata	Wikipedia	Open domain	7,895,789	**4,856,439**	11,373,838	663	96%
GraphNarrative	Freebase	Wikipedia	Open domain	**8,769,634**	1,853,752	**15,472,249**	**1,724**	22%

Table 1: Comparison of graph-to-text datasets

edge that is most frequent in Freebase. All these detected edges form the graph G that pairs with W as an instance (G, W) in the dataset. Note that the simple assumptions in this approach may lead to both false positives and false negatives. In practice, the resulting dataset has solid quality (see detailed assessment in 6.2). Nevertheless, our workflow of dataset creation allows for more advanced and accurate methods in each component.

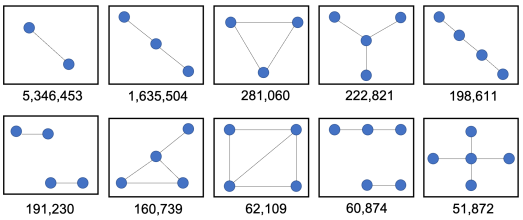

Figure 2: 10 most frequent graph shapes in GraphNarrative, with instance counts

3.2 GraphNarrative Characteristics

This section qualitatively and quantitatively analyzes how GraphNarrative bridges the gap between graph-to-text models and real-world settings for large-scale, open-domain knowledge graphs.

- *Scale and variety of entities and relations*: GraphNarrative contains 8,769,634 graph-sentence pairs, 1,853,752 entities, 15,472,249 triples, and 1,724 relations from 84 Freebase domains (e.g., "people", "book"; complete domain list in Appendix C.1). As Table 1 shows, most other datasets are significantly smaller in these aspects.
- *Linguistic variability*: By using Wikipedia as text corpus, the (graph, text) pairs in GraphNarrative allow a model to learn from many Wikipedia authors' diverse narrations of various relations. On the contrary, text in hand-crafted datasets such as WebNLG tend to follow monotonous templates used by a small number of human contributors.
- *Graph structure complexity*: The graphs in GraphNarrative contain 1–15 triples and 2–20 entities, and they are in a total of 7,920 distinct topological shapes based on graph isomorphism. (Detailed distributions of graph instances and shapes can be found in Appendix C.2.) The 10 most frequent shapes along with their instance counts are shown in Figure 2. Furthermore, only 22% of the instance graphs are star graphs. On the contrary, EventNarrative, TEKGEN are dominated by star graphs, as Table 1 shows.

4 Model

Existing graph-to-text models often use an encoder-decoder structure (Sutskever et al., 2014), where the encoder learns representations of input graphs and the decoder translates the representations into token sequences. These models can be categorized into two groups that differ in graph representations—1) sequence-to-sequence models that encode linearized graphs' token sequences with LSTMs (Trisedya et al., 2018; Gardent et al., 2017b) or Transformers (Castro Ferreira et al., 2019), and 2) models that use a graph encoder to capture the structure of knowledge graphs (Schmitt et al., 2021; Ribeiro et al., 2020; Marcheggiani and Perez-Beltrachini, 2018). Models that attained the state-of-the-art results on WebNLG (Ribeiro et al., 2021; Wang et al., 2021; Clive et al., 2021) belong to the first group. These models fine-tune transformer-based PLMs, e.g., GPT-2 (Radford et al., 2019), BART (Lewis et al., 2020) and T5 (Raffel et al., 2020).

Following the same approach, we fine-tuned T5 and BART on GraphNarrative and other datasets in comparison. In training and applying a graph-to-text model, an instance graph is linearized into a token sequence. Following the method in (Ribeiro et al., 2021), the graph in Figure 1 would be linearized as "`<H> John Douglas <R> place of birth <T> Morgantown, West Virginia <H> John Douglas <R> education institution <T> Tates Creek High School <H> Tates Creek High School <R> location <T> Lexington, Kentucky`" where the special tokens `<H>`, `<R>` and `<T>` denote subjects relations, and objects, respectively.

5 Mitigation of Hallucination

This section focuses on how to mitigate the hallucination problem as explained in Section 1. The culprit of hallucination is fabrication in training

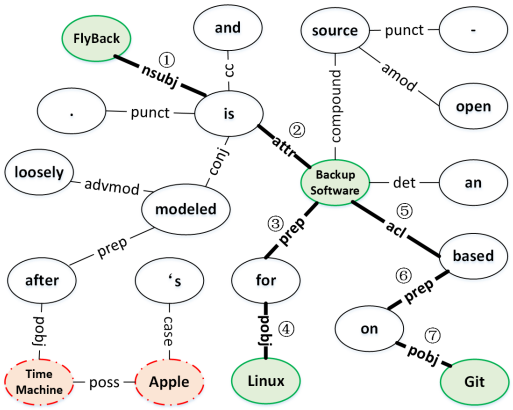

Figure 3: Dependency parse tree (DPT) of the sentence *"FlyBack is an open-source Backup Software for Linux based on Git and modeled loosely after Apple's Time Machine."*

data—textual descriptions containing information not found in input graphs. This is evidenced by that, while graph-to-text models frequently produce hallucination when trained on TEKGEN, it rarely happens on WebNLG. The clean, manually-crafted WebNLG seldom has hallucinated facts in its training data. In contrast, automatically extracted graph-text pairs in TEKGEN contain hallucinated facts due to extraction errors.

There are two directions in tackling graph-to-text hallucination. One direction is to improve our graph-text alignment method (Section 3.1). Given a (graph, text) pair in the result of the alignment, if the graph extracted from the text has missed certain entities and/or relationships in the text, the pair may misguide the trained model to hallucinate facts. A more accurate alignment method can reduce such erroneous pairs and hence reduce hallucinated facts in model output. However, this method has an inherent limitation—a knowledge graph in real-world is far from complete; hence, there will always be facts in text that cannot be mapped to the knowledge graph, i.e., the aforementioned erroneous pairs. Nevertheless, in principle, it could be beneficial to combine this approach with the other approach discussed below, which is an area for future investigation.

The other direction is opposite and is what this study explores. Given a (Freebase subgraph G, Wikipedia sentence W) pair from our alignment method, we introduce a *sentence trimming* algorithm (Algorithm 1) to trim W into a trimmed sentence W_{trim} by eliminating portions that are not present in G while preserving the sentence's main idea. While the detailed algorithm pseudo code and description can be found Appendix A, below we provide its outline.

First, the algorithm parses W and generates its dependency parse tree (DPT) W_{tree}. [2] Then, for each triple $t_i = (s_i, p_i, o_i) \in G$, it identifies the shortest dependency path (SDP) between s_i and o_i, i.e., the shortest path between the two entities' tokens in W_{tree}. It then finds the leftmost position index min_pos in sentence W among all tokens across all triples' SDPs, and similarly the rightmost position index max_pos. This process results in the trimmed sentence W_{trim}, a sub-sequence of W spanning from min_pos and max_pos.

We use the example in Figure 3 to illustrate the algorithm. The figure illustrates the DPT of the sentence W in its caption. The corresponding graph G from the graph-text alignment process is {(`FlyBack`, *software_genre*, `Backup Software`), (`FlyBack`, *operating_system*, `Linux`), (`FlyBack`, *basis*, `Git`)}. Note that entities `Apple` and `Time Machine` in W are missing from G. The SDPs for the three triples are (①, ②), (①, ②, ③, ④), and (①, ②, ⑤, ⑥, ⑦), respectively. Given these SDPs, min_pos is attained by `FlyBack` and max_pos is attained by `Git`. Hence, W_{trim} is `FlyBack is an open-source Backup Software for Linux based on Git`. The sequence `and modeled loosely after Apple's Time Machine.`, related to the missed entities `Apple` and `Time Machine`, is trimmed from W.

Note that, a regular DPT will break up entities such as `Backup Software` into individual tokens, each for a node in the DPT. To avoid that, we used a modified concept of DPT—we preprocessed entity names and tokenized each entity's name into a single token. Hence, the two tokens `Backup` and `Software` were combined into token `BackupSoftware`.

6 Experiments and Results

6.1 Datasets

We performed experiments on three datasets: GraphNarrative, TEKGEN (the large-scale, open-domain graph-to-text dataset that resembles ours the most), and WebNLG (the only graph-to-text dataset with human-annotated text). Statistics about these and other datasets are in Table 1.

- GraphNarrative is partitioned into training, development and test sets. To evaluate the model's generalization capability, the test set is further divided into *seen* and *unseen* partitions. The seen

[2] We used spaCy (Honnibal et al., 2020) for dependency parsing.

partition exclusively includes 56 Freebase domains that are already present in the training set, while the unseen partition encompasses 28 domains that are not in the training set. A full list of the domains are in Appendix C.1. In the seen partition, 90%, 5% and 5% of the instances are allocated for training, development and test, respectively. The instances in the unseen partition were exclusively reserved for the test set.

- In TEKGEN, each instance contains a Wikipedia sentence and a subgraph of Wikidata extracted from the sentence.
- We utilized the standard WebNLG 2017 challenge dataset, which includes instances consisting of a graph generated from DBPedia (Auer et al., 2007) and a human-annotated text with one or multiple sentences describing the graph. The test set is divided into two partitions: *seen*, which contains only DBPedia categories present in the training set, and *unseen*, which covers categories not seen in the training set.

6.2 Human and Automatic Evaluation Metrics

Human Evaluation Metrics: For evaluating the quality of both the GraphNarrative dataset and the sentences generated by models, we particularly focused on assessing whether sentences in the dataset or produced by models fabricate facts that are not in the corresponding graphs narrated by the sentences. To the best of our knowledge, no prior study has quantitatively evaluated the quality of graph-to-text datasets or models with regard to hallucination. Specifically, we define the following four metrics: numbers of *hallucinated entities* (entities not present in the graph but mentioned in the sentence), *missed entities* (entities present in the graph but not mentioned in the sentence), *hallucinated relations* (relations not present in the graph but mentioned in the sentence), and *missed relations* (relations present in the graph but not mentioned in the sentence).

In addition, we also evaluate the quality of sentences using average *grammar errors* per sentence, on a scale of 1-5: 5 (no errors), 4 (one error), 3 (two to three errors), 2 (four to five errors), and 1 (more than five errors).

Automatic Evaluation Metrics: On model-generated sentences, we also report automatic evaluation results using standard natural language generation metrics BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005) and chrF++ (Popović, 2015).

6.3 Evaluation and Experiment Results

6.3.1 GraphNarrative Dataset Quality

Three human annotators were asked to evaluate the quality of the graph-sentence pairs in GraphNarrative. We randomly chose 100 graph-sentence pairs, where each sentence has two versions: the original version and a trimmed version using the algorithm in Section 5. The total 200 pairs were then shuffled. Each human annotator scored all 200 pairs using the metrics discussed in Section 6.2, and their scores are averaged.

Table 3 presents the results. In this and ensuing tables, sentence trimming is denoted ST when applicable. Without sentence trimming, the average numbers of hallucinated entities and relations per instance (i.e., a pair of graph in GraphNarrative and the corresponding sentence) are 1.163 and 1.340, respectively. However, with sentence trimming, these numbers are reduced to 0.306 entities and 0.453 relations. This improvement indicates that sentence trimming enhances the alignment between graphs and sentences.

On the other hand, the graphs in GraphNarrative predominantly adhere to the information present in corresponding sentences, as indicated by the much smaller values of metrics *missed entities* and *missed relations*, both less than 0.1. With regard to grammar, while there is a slight decline in the grammar score under sentence trimming, the difference (4.793 vs. 4.613) is not substantial.

6.3.2 Model Performance on GraphNarrative

We fine-tuned various versions of T5 (small: 60M parameters, base: 220M parameters, and large: 770M parameters) and BART (base: 140M parameters, large: 400M parameters) models on GraphNarrative dataset for 10^6 steps with a batch size of 8 using the Adam Optimizer (Kingma and Ba, 2014) and an initial learning rate of 3×10^{-5}. We utilized a linearly decreasing learning rate schedule without warm-up and set the maximum target text length to 384 tokens. Our implementation was based on the work of (Ribeiro et al., 2021), which adapted PLMs from Hugging Face (Wolf et al., 2019) for graph-to-text generation tasks.

Table 5 shows the automatic evaluation results for different models' performance on the GraphNarrative dataset. Fine-tuning the T5-large model shows the best performance on most of the

Input Graph	w/o sentence trimming	w/sentence trimming
(Arthur Morry, place of death, Brisbane)	Arthur Morry died in Brisbane, Queensland, aged 79.	Arthur Morry died in Brisbane.
(Annapolis Maryland, location contains, US Naval Academy)	During World War II, the US Naval Academy in Annapolis, Maryland was renamed the US Naval Academy in Annapolis, Maryland, and the US Naval Academy in Annapolis, Maryland was renamed the US Naval Academy in Annapolis, Maryland.	US Naval Academy in Annapolis, Maryland.
(Goldie Gets Along, film directed by, Malcolm St. Clair filmmaker) (Goldie Gets Along, film performance actor, Lili Damita) (Goldie Gets Along, film performance actor, Charles Morton actor)	Goldie Gets Along is a 1951 American comedy film directed by Malcolm St. Clair (filmmaker) and starring Lili Damita and Charles Morton (actor).	Goldie Gets Along was directed by Malcolm St. Clair (filmmaker) and starred Lili Damita and Charles Morton (actor).

Table 2: Comparison of generated sentences with and without sentence trimming for selected input graphs

ST	hallucinated entities	missed entities	hallucinated relations	missed relations	grammar
w/o	1.163	0.003	1.340	0.083	4.793
w/	0.306	0.003	0.453	0.040	4.613

Table 3: Human evaluation of GraphNarrative quality

ST	hallucinated entities	missed entities	hallucinated relations	missed relations	grammar
w/o	1.643	0.063	1.363	0.240	4.613
w/	0.260	0.056	0.300	0.370	4.356

Table 4: Human evaluation result of T5-large model generated sentences with/without sentence trimming

metrics, which is consistent with the findings on the WebNLG dataset by (Ribeiro et al., 2021; Wang et al., 2021). To enhance clarity, we refer to the fine-tuned T5-large model on GraphNarrative with sentence trimming as GNST-T5.

6.3.3 Enhancing the Generalization Ability

In order to assess the potential improvement of PLMs' generalization ability through GraphNarrative, we conducted additional fine-tuning of GNST-T5 on the WebNLG benchmark for 100 epochs with an early stopping patience of 20, while keeping other hyperparameters consistent with those in Section 6.3.2. It is worth noting that no trimming was performed on WebNLG, as its sentences were authored by human annotators and exhibited very few hallucinated or missed entities and relations.

Table 6 compares the performance of different graph-to-text models on WebNLG test set. We re-print the results from the seven studies listed in the table. It shows that our GNST-T5 model outperforms others on most metrics, especially on the unseen category, which indicates GraphNarrative improves PLMs' generalization ability.

6.3.4 Ablation Study

We show the effectiveness of sentence trimming in improving model performance on GraphNarrative, TEKGEN, and WebNLG. For GraphNarrative, we fine-tuned the T5 and BART models with and without *sentence trimming* using the same setup in Section 6.3.2. For TEKGEN, we fine-tuned the T5-large and BART-large models with and without sentence trimming using the serialized triples provided by (Agarwal et al., 2021), with the same hyperparameters in Section 6.3.2. For WebNLG, we fine-tuned the T5-large models gotten from fine-tuned on GraphNarrative and TEKGEN with and without ST, respectively, using the same hyperparameters as in Section 6.3.3.

Table 5 and 8 display the automatic evaluation results for BART and T5 models fine-tuned with and without sentence trimming on GraphNarrative and TEKGEN. The metrics (BLEU, METEOR, chrF++) consistently improve with sentence trimming, indicating improved model fit to the trimmed sentences. Trimmed sentences contain less extraneous information not present in input graphs, reducing the likelihood of fabricated entities/relations by PLMs and resulting in improved metrics. Table 7 presents the results on the WebNLG test set, showing that fine-tuned PLMs with sentence trimming consistently outperform those without, highlighting the effectiveness of *sentence trimming* in enhancing PLM performance.

6.3.5 Sentence Trimming for Mitigating Hallucination

We randomly sampled 100 graphs from the test set of GraphNarrative, along with the corresponding generated sentences by the T5-large model, which had been fine-tuned on original Wikipedia sentences and trimmed sentences, respectively. Subsequently, we shuffled the 200 pairs and used three human evaluators to assign scores on the pairs, in the same fashion as described in Section 6.3.1.

Table 4 presents the average evaluation scores assigned by human annotators to sentences generated from a randomly selected set of 100 graphs by T5-large models. The models were fine-tuned using original Wikipedia sentences and trimmed sentences, respectively. The results indicate that the generated sentences exhibit few omissions of enti-

Model	ST	BLEU			METEOR			chrF++		
		all	seen	unseen	all	seen	unseen	all	seen	unseen
BART-base	w/o	33.18	33.33	27.52	17.18	17.26	14.63	36.56	36.74	30.75
BART-base	w/	**46.49**	**46.77**	36.67	24.43	24.53	21.30	49.92	50.12	43.29
BART-large	w/o	32.35	32.48	27.56	17.45	17.53	15.07	37.12	37.29	31.58
BART-large	w/	46.04	46.18	40.98	24.35	24.41	22.17	49.69	49.85	44.72
T5-small	w/o	19.48	19.53	17.34	15.78	15.85	13.79	33.92	34.08	28.90
T5-small	w/	43.72	43.87	38.11	23.40	23.48	21.10	48.15	48.31	42.65
T5-base	w/o	16.89	16.95	14.63	16.23	16.30	14.10	35.37	35.54	29.84
T5-base	w/	42.18	42.29	37.85	24.20	24.27	21.94	49.63	49.80	44.18
T5-large	w/o	22.22	22.26	20.41	17.16	17.23	15.02	36.78	36.95	31.40
T5-large	w/	45.12	45.16	**43.40**	**24.77**	**24.84**	**22.54**	**50.44**	**50.60**	**45.21**

Table 5: Model performance on GraphNarrative

Model	BLEU			METEOR			chrF++		
	all	seen	unseen	all	seen	unseen	all	seen	unseen
(Gardent et al., 2017b)	33.24	52.39	6.13	23.00	37.00	7.00	-	-	-
(Marcheggiani and Perez-Beltrachini, 2018)	55.90	-	-	39.00	-	-	-	-	-
(Ferreira et al., 2019)	51.68	56.35	38.92	32.00	41.00	21.00	-	-	-
(Ribeiro et al., 2020)	-	63.69	-	-	44.47	-	-	76.66	-
(Ribeiro et al., 2021)	59.70	64.71	53.67	44.18	45.85	**42.26**	75.40	78.29	72.25
(Wang et al., 2021)	60.56	66.07	53.87	44.00	46.00	42.00	-	-	-
(Aghajanyan et al., 2021)	56.30	64.80	46.10	42.00	46.00	38.00	-	-	-
GNST-T5 (ours)	**61.46**	**66.49**	**55.35**	**44.30**	**46.23**	42.08	**76.20**	**79.35**	**72.76**

Table 6: Performance comparison of different graph-to-text models on WebNLG test set

Dataset	ST	BLEU			METEOR			chrf++		
		all	seen	unseen	all	seen	unseen	all	seen	unseen
TEK	w/o	60.43	65.49	54.32	44.06	46.04	41.90	75.73	78.83	70.13
GEN	w/	60.82	65.42	55.12	44.25	46.13	42.18	76.16	79.11	72.35
Graph	w/o	60.26	65.44	54.06	44.08	45.90	41.98	75.83	79.02	72.35
Narrative	w/	61.46	66.49	55.35	44.30	46.23	42.08	76.20	79.35	72.76

Table 7: Ablation study: performance of T5-large on WebNLG test set

Model	ST	BLEU	METEOR	chrF++
BART-large	w/o	41.51	23.62	47.13
BART-large	w/	48.32	29.90	57.50
T5-large	w/o	43.03	24.21	48.05
T5-large	w/	**49.83**	**30.52**	**58.25**

Table 8: Performance of BART and T5 on the TEKGEN dataset with and without sentence trimming

ties or relations from the graph, with an average of less than 0.07 missed entities and 0.38 missed relations, both considerably lower than 1. Furthermore, the model trained on trimmed sentences shows a reduction of 1.4 hallucinated entities and 1.0 hallucinated relations, suggesting that sentence trimming effectively mitigates the occurrence of hallucinations. Regarding grammar, sentences generated by the model trained on trimmed sentences show slightly lower scores compared to the model trained on original Wikipedia sentences. However, these scores remain acceptable, with an average number of grammar errors below one.

Table 2 illustrates the input graphs and the sentences generated by T5-large models trained on both the original Wikipedia sentences and trimmed sentences. The model trained on original Wikipedia tends to generate sentences that contain information not present in input graphs. This can include fabricated facts like the age of death of `Arthur Morry` or the *renaming* of the `US Naval Academy` during `World War II`. The third graph example also shows that the model fabricated that Goldie Gets Along is a comedy and was filmed in *1951*, in reality, the movie was filmed in *1933*. Our approach decreases the chances of PLMs fabricating facts by focusing only on the information present in the input graph.

6.3.6 Limitations of Star Graph Datasets

As we stated in Section 1, one problem of existing large-scale knowledge graph-to-text datasets is that they consist of solely star graphs, we investigated the issues arising from such a dataset. Due to space limitations, we put the details about this experiment in Appendix B. The results indicate that a PLM fine-tuned on a dataset consisting solely of star graphs demonstrates poor performance when applied to general graph shapes, which are commonly encountered in real-world applications. Fine-tuning PLMs on a dataset encompassing diverse graph shapes enhances the generalization capability of PLMs.

7 Conclusion

In this paper, we propose a novel approach to mitigating the hallucination problem in natural language generation from large-scale, open-domain knowledge graphs, and we release a large-scale graph-to-text dataset with diverse graph shapes that fills the gap between existing datasets and large-scale real-world settings.

Limitations

This paper presents a method and a new dataset for generating natural language descriptions of knowledge graphs. However, it requires a mapping between the knowledge graph entities and Wikipedia entities. If no such mapping exists, the method cannot be applied. Additionally, the sentence trimming approach proposed in this paper can alleviate the problem of hallucination in PLMs, however, it may result in grammatical errors. Furthermore, the method focuses on describing the content of the input graph without considering context entities/relations in the knowledge graph, which may make the generated sentences less natural. The method does not handle mediator nodes in knowledge graphs, and this is an area of future research. We also need to add a comparison with the approach of further fine-tuning PLMs on the WebNLG dataset regarding reducing hallucinations. The sentence trimming exclusively focuses on trimming the irrelevant portions at the beginning and the end of a sentence, while leaving the subsequences in the middle untouched, this part could also be irrelevant to the overall context.

Ethics Statement

In the course of conducting our research and developing our system, we have striven to remain aware and attentive to the potential ethical implications and challenges. Our ethical considerations were informed by the following:

Potential Threats from Generative AI Given that our research focuses on the task of graph-to-text generation with the aim to produce natural language descriptions of knowledge graphs, we are particularly aware of the potential misuse of our system for misinformation, automated deception, and the generation of biased or unfair content. We have striven to design our system and training methodology, specifically our sentence trimming module, to minimize such potential misuse. The sentence trimming aids in reducing hallucination, or generation of unwarranted or unsubstantiated information. We strongly encourage other developers and researchers to consider these factors when creating similar systems.

Data Privacy and Bias Considerations While our proposed GraphNarrative dataset and future graph-to-text API aim to use publicly available data from sources such as Freebase and Wikipedia, we are conscious of potential issues related to data privacy and bias. Our system operates on anonymized versions of this data, making it impossible to trace back any specific personal identification from the data. We will not store or log user inputs, further securing data privacy.

In addition, we acknowledge that our reliance on Wikipedia data may inadvertently introduce bias, as Wikipedia content can reflect the views of its contributors. We are also aware of the potential for bias in less commonly spoken languages, where the number of contributors might be limited. We have put into place a series of bias mitigation measures, including balanced sampling techniques, and continuous auditing and evaluation of our model outputs for potential bias. We encourage future collaborations with local experts to help identify and rectify such biases.

Ethical Use of Generated Content We recognize that our system's outputs, as they are automated natural language descriptions, can be repurposed in various ways. We firmly urge users and developers to use this content responsibly, with respect to intellectual property rights, and to avoid propagating harmful content or misinformation. Furthermore, we recommend users clearly label AI-generated content, promoting transparency and trust.

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.

Armen Aghajanyan, Dmytro Okhonko, Mike Lewis, Mandar Joshi, Hu Xu, Gargi Ghosh, and Luke Zettlemoyer. 2021. HTLM: Hyper-text pre-training and prompting of language models. *arXiv preprint arXiv:2107.06955*.

Giuseppe Attardi. 2019. Wikiextractor. https://github.com/attardi/wikiextractor. Accessed: Sep 20th, 2019.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: A nucleus for a web of open data. In *Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference*, pages 722–735.

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.

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.

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

Jordan Clive, Kris Cao, and Marek Rei. 2021. Control prefixes for text generation. *arXiv preprint arXiv:2110.08329*.

Anthony Colas, Ali Sadeghian, Yue Wang, and Daisy Zhe Wang. 2021. Eventnarrative: A large-scale event-centric dataset for knowledge graph-to-text generation. In *Proceedings of the 35th Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Andras Csomai and Rada Mihalcea. 2008. Linking documents to encyclopedic knowledge. *IEEE Intelligent Systems*, 23(5):34–41.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2018. Findings of the E2E NLG challenge. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 322–328.

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

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017a. Creating training corpora for NLG micro-planning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 179–188.

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

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform. In *arXiv preprint arXiv:1803.07640*.

Nicolas Heist, Sven Hertling, Daniel Ringler, and Heiko Paulheim. 2020. Knowledge graphs on the web–an overview. *Knowledge Graphs for Explainable Artificial Intelligence: Foundations, Applications and Challenges*, pages 3–22.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spacy: Industrial-strength natural language processing in python.

Nandish Jayaram, Rohit Bhoopalam, Chengkai Li, and Vassilis Athitsos. 2016. Orion: Enabling suggestions in a visual query builder for ultra-heterogeneous graphs. *arXiv preprint arXiv:1605.06856*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jan-Christoph Klie. 2022. wikimapper. https://github.com/jcklie/wikimapper.

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

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on*

Empirical Methods in Natural Language Processing, pages 188–197.

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.

Shiqi Liang, Kurt Stockinger, Tarcisio Mendes de Farias, Maria Anisimova, and Manuel Gil. 2021. Querying knowledge graphs in natural language. *Journal of Big Data*, 8(1):1–23.

Chao Ma, Chunhua Shen, Anthony Dick, Qi Wu, Peng Wang, Anton van den Hengel, and Ian Reid. 2018. Visual question answering with memory-augmented networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6975–6984.

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

Joseph F McCarthy and Wendy G Lehnert. 1995. Using decision trees for coreference resolution. *arXiv preprint cmp-lg/9505043*.

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.

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.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

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 FR Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. 2021. 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.

Leonardo FR 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.

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*, pages 10–21.

Wei Shen, Jianyong Wang, and Jiawei Han. 2014. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering*, 27(2):443–460.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, 27.

Bayu Distiawan Trisedya, Jianzhong Qi, Rui Zhang, and Wei Wang. 2018. GTR-LSTM: A triple encoder for sentence generation from RDF data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 1627–1637.

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

Qingyun Wang, Semih Yavuz, Xi Victoria Lin, Heng Ji, and Nazneen Rajani. 2021. Stage-wise fine-tuning for graph-to-text generation. In *Proceedings of the ACL-IJCNLP 2021 Student Research Workshop*, pages 16–22.

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*.

Yikun Xian, Zuohui Fu, Shan Muthukrishnan, Gerard De Melo, and Yongfeng Zhang. 2019. Reinforcement knowledge graph reasoning for explainable recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 285–294.

Gensheng Zhang, Damian Jimenez, and Chengkai Li. 2018. Maverick: Discovering exceptional facts from knowledge graphs. In *Proceedings of the 2018 ACM SIGMOD International Conference on Management of Data*, pages 1317–1332.

Jiaping Zheng, Wendy W Chapman, Rebecca S Crowley, and Guergana K Savova. 2011. Coreference resolution: A review of general methodologies and applications in the clinical domain. *Journal of Biomedical Informatics*, 44(6):1113–1122.

Li Zhou and Kevin Small. 2019. Multi-domain dialogue state tracking as dynamic knowledge graph enhanced question answering. *arXiv preprint arXiv:1911.06192*.

A Algorithm

Algorithm 1 shows the details of sentence trimming. Lines 1—6 are to ensure that an entity consisting of multiple tokens is tokenized into one single token, and the mapping M is to recover the entities' tokens in W to get W_{trim}. Lines 17—18 are the recovery step. Lines 9—16 are the process of finding the leftmost position min_pos and rightmost position max_pos from W by scanning each triple (s, p, o) in G and finding the tokens on SDPs. $node$ denotes a token on the SDP of W_{tree} between entity s and o. $node.start$ and $node.end$ denotes $node$'s starting position index and ending position index in W, respectively. $node.start$, $node.end$, min_pos, and max_pos are on character level. If $node$ appears multiple times in W, $node.start$ will be the start index of the first one and $node.end$ will be the end index of the last one.

Algorithm 1: Sentence Trimming

Sentence Trimming *(G, W)*
Input: W: A co-reference resolved Wikipedia sentence
G: The aligned graph from $\mathcal{G}$ for W based on Graph-Text Alignment
Output: W_{trim}: The trimmed Wikipedia text sequence
1 $M \leftarrow \{\}$
2 **foreach** $(s, p, o) \in G$ **do**
3 $s' \leftarrow s.remove(special\_tokens)$;
4 $o' \leftarrow o.remove(special\_tokens)$;
5 $W \leftarrow W.replace((s, o), (s', o'))$;
6 $M[s'], M[o'] \leftarrow s, o$;
7 $W_{tree} \leftarrow W.dependency\_parsing()$;
8 $min\_pos, max\_pos \leftarrow W.length, 0$;
9 **foreach** $(s, p, o) \in G$ **do**
10 $sdp \leftarrow shortest\_path(W_{tree}, s', o')$;
11 **foreach** $node \in sdp$ **do**
12 **if** $node.start < min\_pos$ **then**
13 $min\_pos \leftarrow node.start$;
14 **if** $node.end > max\_pos$ **then**
15 $max\_pos \leftarrow node.end$;
16 $W_{trim} \leftarrow W[min\_pos : max\_pos]$;
17 **foreach** $key \in M$ **do**
18 $W_{trim} \leftarrow W_{trim}.replace(key, M[key])$;
19 **return** W_{trim}

B More Experiment Details

B.1 Limitations of Star Graph Datasets

Star and Non-Star Split To investigate the impact of star and non-star graphs on the model's performance, we differentiate graph-sentence pairs into *star* instances (instances with star graphs) and *non-star* instances (instances with non-star graphs). We excluded instances with fewer than three entities, as they could be considered as both path and star shape. Table 9 provides statistics regarding the distribution of star and non-star graphs in GraphNarrative. The number of non-star instances in all three sets is approximately 3.5 times greater than that of star instances. In order to eliminate this inequality and ensure a fair comparison, we randomly selected an equal number of star and non-star instances for each of the three sets, which resulted in the following numbers: 290,047 instances for training, 16,192 instances for development, and 16,425 instances for testing.

	star	**non-star**	**all**
train	290,047	290,047 (out of 1,022,303)	580,094 (out of 1,312,350)
dev	16,192	16,192 (out of 56,612)	32,184 (out of 72,804)
test	16,425	16,425 (out of 58,517)	32,850 (out of 74,942)
all	322,664	322,664 (out of 1,137,432)	645,328 (out of 1,460,096)

Table 9: Number of star and non-star instances of GraphNarrative dataset used in experiments about graph shapes

As we stated in Section 1, one problem of existing large-scale knowledge graph-to-text datasets is that they consist of only star graphs, we investigated the difference in model performance if we train a model using star-only (star), non-star-only (non-star), and both star and non-star (both) graphs.

We fine-tuned T5-large model with (original) and without (trimmed) sentence trimming on star, non-star, and both datasets got in Section 3 for 10 epochs with early stopping patience 5, with the same hyperparameters as in Section 6.3.2.

Table 9 presents the performance of the model on star, non-star, and combined datasets. On the test set, the model trained specifically on a particular type of graph shape exhibited the highest performance on that corresponding shape type. For instance, the model trained on star shapes performed best on the star dataset, while the model trained on non-star shapes performed best on the non-star dataset. Besides, the model trained on star instances and tested on non-star instances performs worse compared to the model trained on non-star instances and tested on star instances. The results indicate that a PLM fine-tuned on a dataset consisting solely of star graphs demonstrates poor performance when applied to general graph shapes, which are commonly encountered in real-world applications. Fine-tuning PLMs on a dataset encompassing diverse graph shapes enhances the generalization capability of PLMs.

Sentence	Train	Test	BLUE	METEOR	chrF++
original	star	star	36.57	22.75	46.70
		non-star	30.64	21.89	45.48
		both	33.54	22.32	46.09
	non-star	star	34.02	21.67	44.54
		non-star	37.18	23.99	50.02
		both	35.71	22.99	47.57
	both	star	36.23	22.94	47.06
		non-star	36.86	24.32	50.61
		both	36.56	23.63	48.83
trimmed	star	star	47.70	26.62	52.08
		non-star	37.75	25.17	50.20
		both	42.48	25.88	51.14
	non-star	star	45.83	25.72	50.33
		non-star	47.30	27.73	55.21
		both	46.61	26.73	52.77
	both	star	47.60	26.87	52.52
		non-star	46.83	27.89	55.45
		both	47.20	27.38	53.99

Table 10: Model performance on star and non-star graphs

C More Details about GraphNarrative

C.1 List of Domains

Seen domains: symbols, religion, amusement_parks, time, automotive, meteorology, award, biology, comic_strips, tennis, event, boats, internet, sports, location, broadcast, business, measurement_unit, finance, law, metropolitan_transit, astronomy, tv, fictional_universe, royalty, computer, media_common, aviation, language, baseball, protected_sites, olympics, geography, book, medicine, influence, spaceflight, basketball, comic_books, theater, transportation, government, music, soccer, visual_art, education, people, military, organization, travel, american_football, architecture, cvg, film.

Unseen domains: martial_arts, games, ice_hockey, cricket, rail, food, opera, projects, dining, skiing, conferences, library, exhibitions, zoos, boxing, engineering, digicams, venture_capital, chemistry, celebrities, chess, interests, distilled_spirits, comedy, fashion, geology, wine, bicycles.

C.2 GraphNarrative Characteristics

#entities	2	3	4	5	6	7	8	9	10	11
#shapes	1	2	7	23	122	705	1690	1705	1267	830
#entities	12	13	14	15	16	17	18	19	20	sum
#shapes	542	378	222	176	106	58	52	22	12	7920

Table 11: Distribution of GraphNarrative graph shapes by number of entities

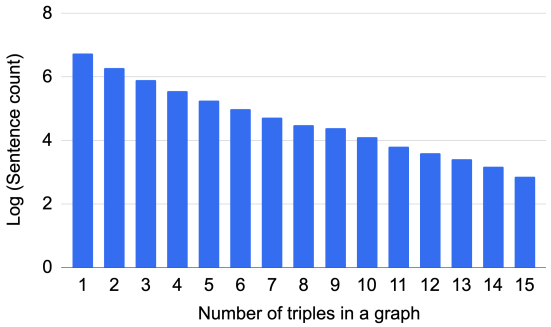

Figure 4: Distribution of GraphNarrative instances by number of triples in graphs

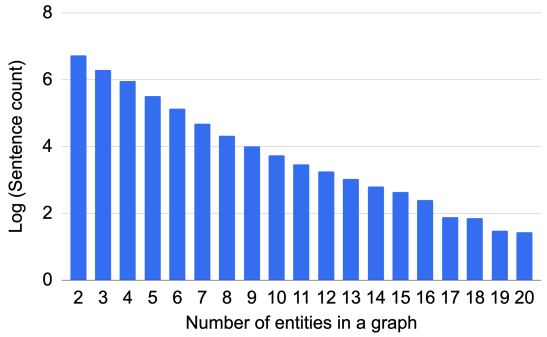

Figure 5: Distribution of GraphNarrative instances by number of entities in graphs

C.3 Dataset Creation

C.3.1 Pre-processing of Text Corpus and Knowledge Graph

Our text corpus is derived from the Wikipedia dump [3] which was released on Sep. 1st, 2019. In preprocessing the corpus, we used WikiExtractor (Attardi, 2019) to transform the raw Wikipedia dump in compressed XML file into numerous plain text files containing bodies of Wikipedia articles without tables, infoboxes, table of contents, categories, and so on. [4]

We utilized the most recent Freebase dump as our knowledge graph source. [5] Most relations in the graph form semantically-redundant reverse pairs. If the input graph triples to a graph-to-text model containing such reverse edges, we only need to simply retain one edge out of each redundant pair. Hence, we did exactly that in pre-processing the whole Freebase dump so that our input graphs will not have reverse edges. Furthermore, our pre-processing also got rid of the mediator (CVT) nodes (Bollacker et al., 2008) from the Freebase dump by concatenating edges connected through mediator nodes.

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

[4] https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Layout

[5] https://developers.google.com/freebase

C.3.2 Graph-Text Alignment

Wikipedia-to-Freebase entity mapping. We collected a Wikipedia-to-Freebase entity mapping that serves as an intermediary bridge between Wikipedia titles and their corresponding Freebase entities. The Wikipedia-to-Freebase entity mapping maps 4,408,115 English Wikipedia titles to their corresponding Freebase entities.

To cover as many accurate Freebase entities as possible, we employed three methods to map Wikipedia titles to Freebase entities: 1) parsing the Freebase data dump to get the Wikipedia-to-Freebase entity mapping using https://github.com/saleiro/Freebase-to-Wikipedia, 2) inferring from the Wikipedia-to-Wikidata using wikimapper (Klie, 2022) and Wikidata-to-Freebase mapping dumps https://developers.google.com/freebase, and 3) inferring from the Wikipedia-to-DBpedia and DBpedia-to-Freebase mapping dumps http://downloads.dbpedia.org/2016-10/core-i18n/en/ "freebase links". By combining all of the above dumps and eliminating conflicting entity mappings, we were able to map 4,408,115 English Wikipedia titles to their corresponding Freebase entities. The Wikipedia-to-Freebase entity mapping file link can be found at our Anonymous GitHub mentioned in the Abstract.

Coreference resolution. To enhance the coherence and readability of the Wikipedia sentence W, coreference resolution (Zheng et al., 2011) is applied to replace all the Wikipedia token spans with the same entities they refer to. We used AllenNLP's coreference resolution (Gardner et al., 2017; Lee et al., 2017) with default settings. We evaluated its performance by randomly selecting 20 Wikipedia articles and assessing the quality of the generated coreference resolution results. Based on a human evaluation, there were 689 entities that were successfully resolved, with 630 being correct and 11 entities that should have been resolved but were missed by the model. This results in a precision of around 91% and recall of around 98%.

C.4 The 41 Shapes in WebNLG

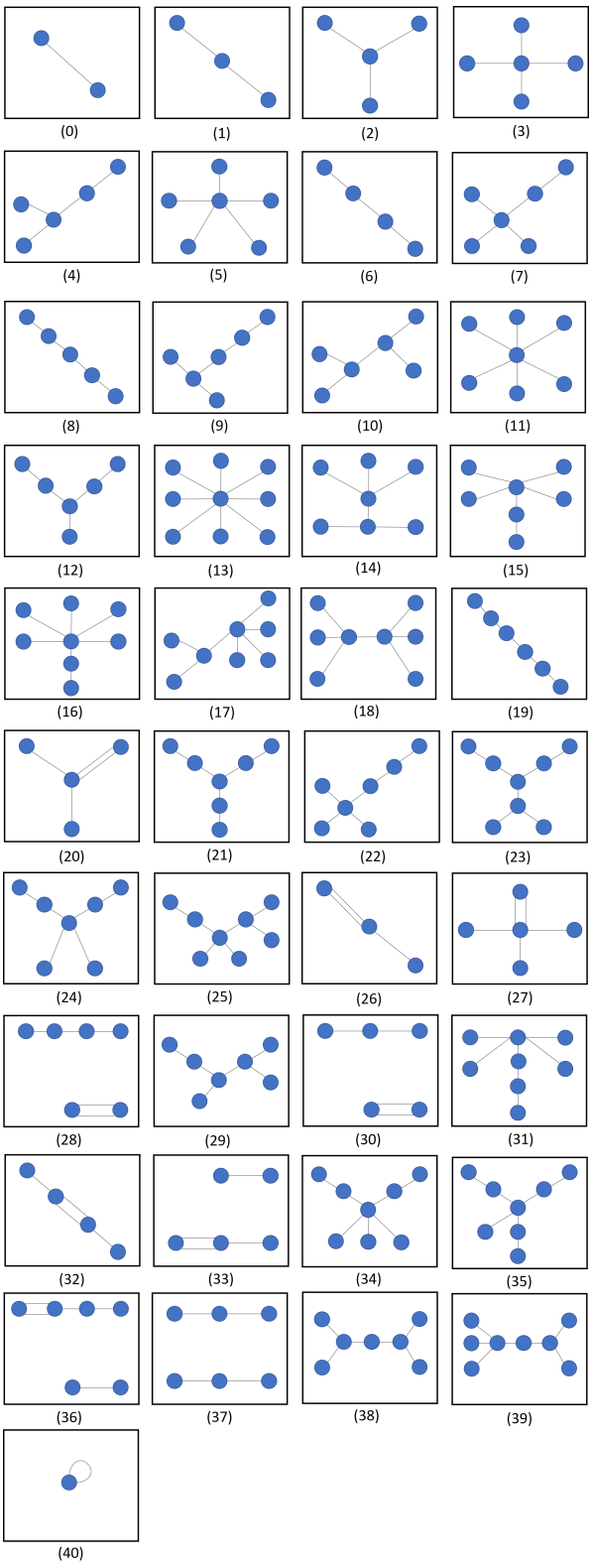

Figure 6: WebNLG shapes