NATURAL LANGUAGE GENERATION FROM LARGE-SCALE OPEN-DOMAIN KNOWLEDGE GRAPHS

by

XIAO SHI

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

Aug 2024

To my family.

ACKNOWLEDGEMENTS

I express my deep gratitude to my advisor, Dr. Chengkai Li, for his invaluable guidance, training, and care throughout my entire Ph.D. journey. I began my Ph.D. studies with a limited background in computer science, having earned my bachelor's degree in Electronic Information Engineering without taking core computer science courses such as Algorithms, Compilers, Computer Networks, and Operating Systems. Despite this, Dr. Li never pushed me but allowed me to progress at my own pace. My interest in natural language processing and machine learning research was sparked after taking Dr. Li's CSE5334 Data Mining course. His keen interest and enthusiasm in teaching, along with his effective pedagogical methods, truly inspired me. The same enthusiasm that he brought to teaching was evident in his advising, helping to transform me into a dedicated researcher. Dr. Li has devoted a tremendous amount of time to training me in research, writing, and delivering professional talks. His brilliance and insight in research discussions have been invaluable. He taught me to aim high and strive for perfection in my work. His dedication, persistence, and sincerity in research have deeply impressed me and set high standards that I aspire to maintain throughout my career. I also appreciate Dr. Li's tolerance of my stubbornness; although we sometimes had differing opinions, I often found that he was correct. Dr. Li's influence extends beyond my career to my life attitude. From him, I learned the importance of taking our work seriously and persisting in doing what we believe is right, regardless of external pressures.

I am profoundly grateful to my committee professors, Dr. Upendranath Chakravarthy, Dr. Gautam Das, and Dr. Won Hwa Kim, for their invaluable insights, dedicated time,

and support throughout my Ph.D. journey. Their guidance during my diagnostic evaluation, comprehensive exam, proposal, and defense was instrumental in shaping my research and academic growth. A special thank you to Dr. Kim for providing me with access to his lab's servers, which were crucial for the completion of my experiments. His support was pivotal in enabling my research to progress smoothly. I deeply appreciate the commitment and encouragement from each of my committee members, which have significantly contributed to my development as a researcher.

I owe many thanks to my knowledgeable and caring mentors Ping Liu, Qianqi (Kay) Shen, and Nikita Zhiltsov, and my humorous and supportive manager Jianqiang (Jerry) Shen for the fantastic internship experiences I had at the Talent Marketplace Data Foundation team at LinkedIn. I was fortunate to return to the same team and start my first job, I am immensely thankful to my team members Benjamin Le, Declan Boyd, Dean Young, Ran Zhou, Rajat Arora, Dan Liu, Sicong Kuang, Yunxiang Ren, Michael Dong, Chengming Jiang, Lucky Wang, Yeou Chiou, Yuchin Juan, and Yuan Yin for their invaluable help and support. Upon the submission of this dissertation, this team was divided in the reorganization and I will move to a new team, but the experiences and camaraderie I shared with the TMDF team will always remain a treasured part of my life.

I want to express my heartfelt thanks to my incredible lab mates who created a pleasant and supportive atmosphere in the lab. I am grateful to Zhengyuan Zhu, Zeyu Zhang, Haiqi Zhang, Nasim Shirvani Mahdavi, Mohammed Samiul Saeef, Yogesh Gurjar, Damian Jimenez, and Gensheng Zhang for their excellent collaboration and assistance with my projects, as well as for the memorable moments and support in life. Special thanks to Theodora Toutountzi for her invaluable help with my GHC application, slides, presentations, and for always being supportive. Foram Pankajbhai Patel, thank you for being a fun friend and for all the conversations we shared. I also

appreciate the support from Fatma Arslan and Farahnaz Akrami. I am grateful to Jacob Daniel Devasier, Ishan Poudel, Abhishek Divakar Goudar, Sarbajit Roy, and Prabin Lamichhan for the positive interactions. Because of them, the IDIR lab felt like home to me. Thank you all for always supporting me and for making my Ph.D. journey a rich and fulfilling experience.

I am grateful to my friends who made my Ph.D. journey truly memorable. I would like to extend my heartfelt thanks to Xin Miao for motivating and helping me with preparing my PhD application, assisting me in settling down when I first arrived in the U.S., guiding me in research and career decisions, and encouraging me to find an internship; Lu Zhang for being an incredible friend, for pushing and helping me in my research, and for the enjoyable conversations and hangouts; Xin Ma for assisting me with using GPUs in Dr. Kim's lab and sharing valuable information about Ph.D. milestones; Jie Han, my roommate, for taking care of me when I had COVID; Qiaowen (Jenny) Chen and Daniel Obembe for their help in daily life and for the funny and enjoyable chats; Chaochao Yan and Yong Zhao for their assistance in understanding technical details and their career advice; Jessica Ren, Lu Chen, Grace Li, Mei Liu, Tong Feng, Yan Lin, and Marshell Mu for their unwavering support during difficult times; my neighbors Yanjin (Chrisa) Chen, Cong Chen, and Tingting Xuan for their support in my life; Mengfei Ren, Feng Tong, Kexin Ding, Yujing Yang, Zehao Ye, and Ce Bian for the fun times we shared together; and Chunyuan Li for adopting Guanguan, the cat I love but couldn't adopt in my apartment. Thank you all for being a part of this incredible journey.

I would like to extend my gratitude to the CSE department at UTA and Dr. Li for providing me with financial support throughout my entire Ph.D. My sincere thanks go to Ginger Dickens, Sherri Gotcher, Samantha Oliva, and Nova Coates for their assistance with paperwork, scheduling Ph.D. milestones, and handling financial doc-

uments. I am also grateful to Skipper Harris, Bito Irie, and Edward Orcutt for their help with server and laptop technical issues.

I would like to thank Susan Ball, my international student advisor, for her invaluable assistance with immigration questions and the preparation of CPT and OPT documents.

Finally, my deepest and heartfelt appreciation goes to my family for their unwavering support during my Ph.D. I am grateful to my parents for their encouragement, sacrifice, and patience, which kept me going. I also want to thank my brother and grandparents for their immense love. I would like to thank my cat, Mimi Shi, for his companionship and the joy he brought into my life. I hold a deep memoriam for my grandfather and grandmother who passed away during my Ph.D. journey.

Aug 2024

ABSTRACT

NATURAL LANGUAGE GENERATION FROM LARGE-SCALE OPEN-DOMAIN KNOWLEDGE GRAPHS

XIAO SHI, Ph.D.

The University of Texas at Arlington, 2024

Supervising Professor: Dr. Chengkai Li

This dissertation delves into the realm of natural language generation (NLG) from expansive open-domain knowledge graphs, aiming to bridge the gap between existing methods primarily tested on limited datasets and the demands of real-world large-scale, diverse graph structures. Prior works in NLG often relied on small-scale or restricted datasets, neglecting the complexities of broader knowledge graphs. To address this, we introduce a new dataset called GraphNarrative, designed to encompass a wide range of graph structures and enhance the realism of NLG tasks.

The core contribution of this research lies in devising a novel approach to mitigating information hallucination, a common issue in NLG where generated text may include inaccuracies or fabricated details not present in the input graph. Our method leverages Transformer-based pre-trained language models fine-tuned on GraphNarrative. Notably, we employ dependency parse trees to trim training sentences, ensuring they strictly adhere to the information present in their corresponding graphs.

Through rigorous experimentation and evaluation, we demonstrate the effectiveness of our approach in eliminating information hallucination while maintaining

high-quality NLG output. Our findings showcase significant improvements over existing methods, particularly when applied to diverse and large-scale knowledge graphs.

Furthermore, we contribute to the research community by releasing the GraphNarrative dataset, along with our source code and trained models, available for public access at `https://github.com/idirlab/graphnarrator`.

In conclusion, this dissertation not only advances the field of NLG by addressing challenges posed by large-scale open-domain knowledge graphs but also provides valuable resources and methodologies for future research in this domain.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Natural language generation (NLG) [Bengio et al., 2000] has emerged as a critical area in artificial intelligence, enabling machines to generate human-like text and narratives from structured data sources. For example, in the field of journalism, NLG systems can generate news articles and summaries from raw data, such as financial reports and sports statistics, which allows for real-time updates and content creation without human intervention [Leppänen et al., 2017]. In the healthcare sector, NLG is used to produce patient reports and medical summaries from electronic health records, and this enables streamlined clinical workflows and improves patient care [Cawsey et al., 1997]. Another application of NLG systems is in customer service, where they create personalized responses and assistive texts by analyzing user queries and historical interaction data. This improves the overall user experience and operational efficiency [Golchha et al., 2019]. Additionally, NLG is employed in creative writing, where it assists authors by generating plot suggestions, character dialogues, and entire narrative structures from predefined datasets and templates, showcasing its potential in augmenting human creativity [Alsharhan, 2022]. These examples highlight the versatility of NLG in transforming structured data into coherent, contextually relevant, and human-like text across various industries.

One key challenge in NLG research is the generation of coherent and informative text from large-scale open-domain knowledge graphs. Knowledge graphs represent real-world entities and their relationships, providing a rich source of information for NLG tasks. However, existing NLG models often struggle [Agarwal et al., 2021]

to handle the complexity and diversity of such knowledge graphs, leading to issues like information hallucination where generated text may include inaccurate or fabricated details. This dissertation addresses this challenge by proposing novel techniques specifically designed to mitigate information hallucination and improve the overall reliability and informativeness of NLG systems based on large-scale open-domain knowledge graphs.

The task of *graph-to-text generation* aims to automatically produce 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 $G \subset \mathcal{G}$, generating a token sequence $(y_1, ..., y_n)$ 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, as the outcome of some upstream operation, e.g., 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, Li and Jagadish, 2014] as well as systems backed by knowledge graphs for question-answering [Zhou and Small, 2019, Ma et al., 2018, Bordes et al., 2015] and fact discovery [Xian et al., 2019, Zhang et al., 2018, Lin et al., 2018]. In these places (illustrated in Figure 1.1), 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 still be complex and thus challenging to comprehend.

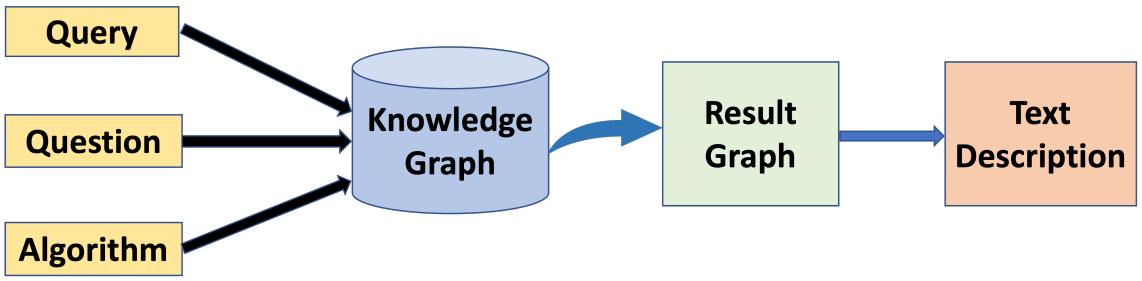


Figure 1.1: Application scenarios of graph-to-text

There are multiple ways to present graphs to users, each with its own drawbacks. Nested dictionaries [Gouws, 2001] can represent graphs but are complex and difficult to interpret [Lee and Lee, 1995]. Resource Description Framework (RDF) [World Wide Web Consortium] triples offer a flattened representation but still pose challenges in understanding the connections and overall information [Chernenkiy et al., 2017]. Visualization [Nararatwong et al., 2020] provides an intuitive way to capture graph structures, but can be time-consuming and impractical for users to understand given large or complex graphs, and unsuitable for systems that require natural language results [Speretta and Gauch, 2005, Zheng, 2002], audio devices such as Alexa ([FitzGerald et al., 2022]) and Siri ([Kaplan and Haenlein, 2019]), or visually impaired users. Instead, presenting graphs in natural language can be sufficient for some use scenarios and help end users understand them better. More detailed discussions regarding this can be found in Chapter 2.

In graph-to-text generation, the preciseness and naturalness of the textual narration of graph fragments is 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 *handcrafted* and *monotonous*, e.g., WebNLG [Gardent et al., 2017b], or only include *simple, special* formations in narrated input fragments, e.g., EventNarrative [Colas et al., 2021] and TEK-

GEN [Agarwal et al., 2021]. Existing graph-to-text models, being trained and evaluated on these datasets, are largely not validated for more realistic large-scale, open-domain settings. Chapter 3 presents this analysis in detail.

This dissertation's research 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 relationships described in the sentence. The large-scale of both Wikipedia and Freebase, the linguistic variation 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 only 22% of the 8.7 million are star graphs, in contrast to 94% and 96% in EventNarrative and TEKGEN, respectively. Chapter 4 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, we adopt the same approach (more details in Chapter 5). As pointed out in [Agarwal et al., 2021] and [Dušek et al., 2018], though, this approach may suffer from information *hallucination*, i.e., the output texts may contain fabricated facts 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] and Ma et al. [2022].

The first two studies attempted to address hallucination by further fine-tuning PLMs on WebNLG after fine-tuning on noisier automatically-extracted datasets. Ma et al. [2022] adopted a different approach, by filtering out training instances when the ROUGE-1 [Lin, 2004] scores between the input and the output fall below a certain threshold. However, these studies did not quantify the prevalence of hallucination in their models' outputs. Nor did they provide direct experiment results or other evidence to verify the approach in reducing hallucination. 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 not represented in the graph. This process, named *sentence trimming*, is accomplished by analyzing the shortest paths between graph entities within the sentence's dependency parse tree (details in Chapter 6).

We conducted comprehensive automatic and human assessments of text descriptions generated by fine-tuned PLMs, specifically BART [Lewis et al., 2020], T5 [Raffel et al., 2020], and TinyLLaMA [Zhang et al., 2024]. The automatic evaluation results consistently demonstrated that models performed better with the use of sentence trimming, across the datasets of GraphNarrative, TEKGEN, WebNLG, and DART [Nan et al., 2021]. The approach led to the increment of 12 and 7 points in BLEU score [Papineni et al., 2002] for GraphNarrative and TEKGEN, respectively. A T5-large model fine-tuned on GraphNarrative with sentence trimming achieved state-of-the-art results on the WebNLG benchmark. Furthermore, human evaluation results showed that sentence trimming on average reduced 1.4 entity hallucinations and 1 relationship hallucination per text description.

Below we briefly summarize our contributions. In this dissertation, we focus on advancing the state-of-the-art in graph-to-text generation by addressing the challenges posed by large-scale open-domain knowledge graphs. We begin by reviewing prior works in graph-to-text generation, which predominantly relied on small-scale, human-annotated datasets or datasets with limited graph structures, such as star graphs. While these datasets were valuable for initial graph-to-text generation research, they failed to capture the full spectrum of complexities present in real-world knowledge graphs. This limitation motivated us to develop the new dataset, GraphNarrative, which encompasses diverse graph structures and provides a more realistic testbed for graph-to-text models.

Our main contribution lies in pioneering the quantification of hallucinations produced by graph-to-text models and proposing a novel approach to mitigating information hallucination in graph-to-text generation. We leverage Transformer-based pre-trained language models fine-tuned on the GraphNarrative dataset, enhancing their ability to generate accurate and informative text from knowledge graphs. Crucially, we utilize dependency parse trees to trim training sentences which helps ensure that, in the training set, output sentences strictly adhere to the information present in input graphs. This approach not only improves the quality of NLG output but also enhances the interpretability and trustworthiness of generated text.

Throughout this dissertation, we explored data, produced statistics, and conducted extensive experiments and evaluations to validate the effectiveness of our approach. We compared our method against existing NLG models trained on both GraphNarrative and other datasets, demonstrating superior performance in terms of reducing information hallucination while maintaining high-quality text generation.

In summary, this dissertation contributes to the advancement of NLG by addressing the challenges posed by large-scale open-domain knowledge graphs. Par-

ticularly, it improves the accuracy and reliability of NLG models. Additionally, we released the GraphNarrative dataset, [1] along with our source code and trained models, [2] to contribute to the NLG research community and facilitate further advancements in this field.

[1] https://zenodo.org/records/12908748

[2] https://github.com/idirlab/graphnarrator

CHAPTER 2

BACKGROUND

2.1 Knowledge Graph

A knowledge graph (KG) is a network of interconnected entities, such as objects, events, or concepts, along with the relationships between them. These entities are represented as nodes, while the relationships are depicted as edges connecting the nodes. Figure 2.1 illustrates a knowledge graph fragment that represents information about strategy game `Civilization` as well as its designer, publisher, subject, origin, derivative, and so on. For example, for the two nodes `Civilization` and `Avalon Hill`, the relationship between them is that `Avalon Hill` is the publisher of the game `Civilization`.

Knowledge graphs are typically stored in the standard Resource Description Framework (RDF) [Pan, 2009] format, which facilitates efficient querying and manipulation using SPARQL (SPARQL Protocol and RDF Query Language) [Pérez et al., 2009]. Databases such as OpenLink Virtuoso [Erling and Mikhailov, 2009] are optimized for storing RDF triples and executing SPARQL queries, making them ideal for applications where RDF and SPARQL are the primary means of interaction. Graph databases, such as Neo4j [Miller, 2013], Amazon Neptune [Bebee et al., 2018] and Microsoft Azure Cosmos DB [Guay Paz and Guay Paz, 2018], are specifically designed to handle graph structures. They offer efficient storage and querying capabilities that are optimized for graph operations. For large-scale knowledge graphs, distributed storage solutions such as Google Bigtable [Chang et al., 2008] and Apache

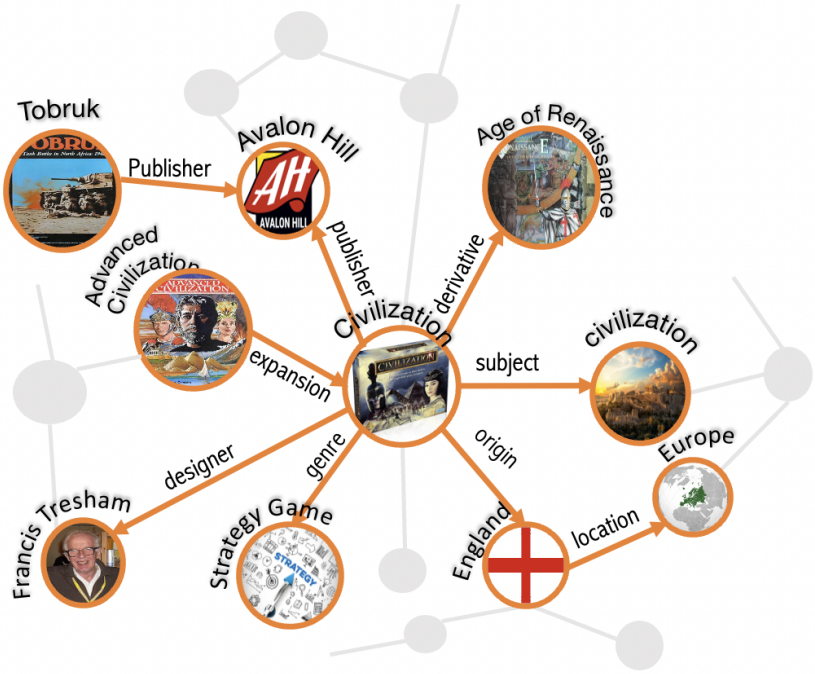


Figure 2.1: A knowledge graph

HBase [Vora, 2011] offer horizontal scalability that can support the management of vast amounts of data distributed across multiple servers.

Knowledge graphs are essential in numerous applications, including search engines, recommendation systems, and artificial intelligence. The primary importance of knowledge graphs lies in their ability to integrate data from diverse sources, enabling better analysis and decision-making. By enhancing the representation of complex relationships between data points, knowledge graphs improve search capabilities with advanced, context-aware querying. Additionally, they facilitate interoperability across different platforms by providing a common semantic framework. This allows machines to process large volumes of information more efficiently, which results in enhanced understanding, improved analytics, and the creation of more intelligent systems and applications.

Knowledge graph-based systems return graphs as results. For instance, a graph query system [Liang et al., 2021, Jayaram et al., 2016, Li and Jagadish, 2014] retrieves a set of subgraphs from a knowledge graph that satisfy the query conditions. A KG-based question answering (QA) system [Zhou and Small, 2019, Ma et al., 2018, Bordes et al., 2015] searches the KG and returns one or more subgraphs as answers to a question. Additionally, certain data mining systems [Xian et al., 2019, Zhang et al., 2018, Lin et al., 2018] discover facts from the knowledge graph using data mining algorithms. These diverse systems leverage the structured nature of knowledge graphs to provide meaningful insights and answers. Consequently, the representation of these result graphs becomes crucial for users to effectively interpret the information.

There are various ways to present result graphs to users. They can be presented using nested dictionaries [Zhang et al., 2018], with nodes as the keys and edges as values. Figure 2.2 shows using nested dictionaries to present the graph in Figure 2.1. Though easy for programming language to process, nested dictionaries (typically in JSON format) have complex hierarchical structures. It is challenging for users to understand the structure and how the nodes and edges are connected.

```
"Advanced Civilization":
        {“games.game.game_subjects”: “Civilization (subject)”,
        {“games.game_expansion.game”: “Civilization” {
                "games.game.publisher": "Avalon Hill",
                "games.game.genre": "Strategy game",
                "games.game.origin": "England",
                “games.game.derivative_games”: “Age of Renaissance” :
                                        {"games.game.publisher": "Avalon Hill"},
        "games.game.publisher": "Avalon Hill",
        "games.game.genre": "Strategy game"}}
```

Figure 2.2: Presenting a graph using dictionaries

A result graph can also be represented using triples, as illustrated in Figure 2.3, which presents the knowledge graph from Figure 2.1. This method offers a flattened view of the graph. Each triple indicates one relationship between two entities. However, as the number of triples increases, it becomes challenging to interpret how the nodes and edges are connected and what overall information it conveys.

```
(Age of Renaissance, games.game.publisher, Avalon Hill)
(Civilization, games.game.game_subjects, Civilization (subject))
(Advanced Civilization, games.game_expansion.game, Civilization)
(Civilization, games.game.publisher, Avalon Hill)
(Advanced Civilization, games.game.publisher, Avalon Hill)
(Civilization, games.game.genre, Strategy game)
(Advanced Civilization, games.game.genre, Strategy game)
(Civilization, games.game.origin, England)
(Civilization, games.game.derivative_games, Age of Renaissance)
```

Figure 2.3: Presenting a graph using triples

Another approach to presenting a result graph is to visualize its nodes and edges. For instance, Figure 2.4 illustrates this by depicting the knowledge graph shown in Figure 2.1. However, such visualization could be time-consuming to interpret when the graph is large or complex. Furthermore, its edge labels can be challenging to comprehend due to their complex structure and technical terminology (e.g., *architecture.building_complex_function.building_complexes*). Additionally, the direction of an edge label, such as *games.game_expansion.game* can lead to confusion, since it is unclear whether `Advanced Civilization` is an expansion of `Civilization` or `Advanced Civilization` is expanded to `Civilization` as it may not be immediately clear how the entities are related based on the label's orientation and wording.

In some cases, visualization may be inapplicable or unsuitable. For example, systems designed to produce natural language results, such as chatbots, do not ben-

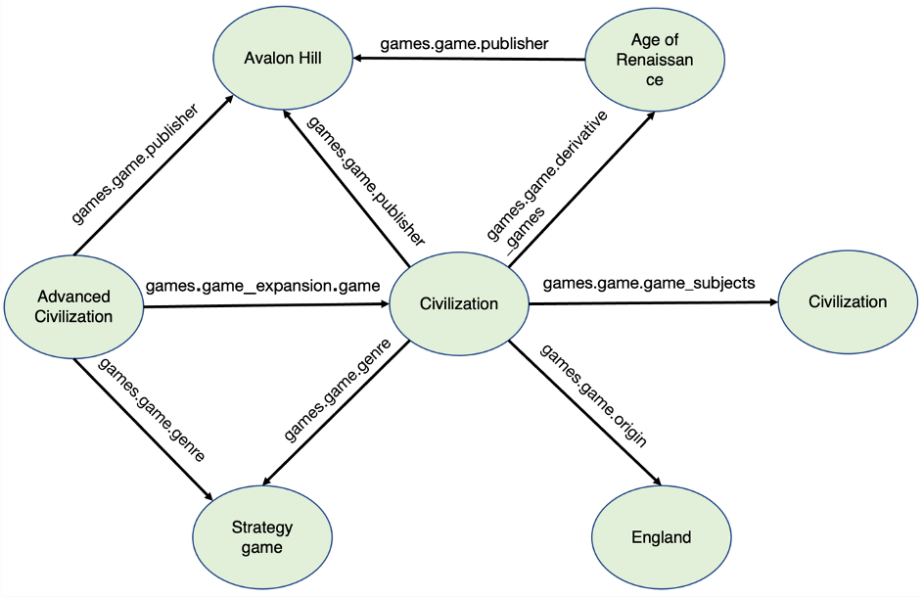


Figure 2.4: Presenting a graph by illustration

efit from graphical displays. Similarly, visualization is not feasible for devices that rely solely on audio output. Additionally, graphical representations pose challenges for visually impaired users, making visualization an ineffective method in related contexts.

For aforementioned scenarios, providing a textual description of a knowledge graph fragment can enable users to swiftly grasp the key information. we are suddenly switching from general discussion to specific example, without proper transition. For instance, given the graph in Figure 2.4, a manually crafted description facilitates a clearer and faster understanding of the information it contains: "Civilization is an England-originated strategy game on the subject of civilization. It has a sequel Age of Renaissance and an expansion Advanced Civilization. All three games were published by Avalon Hill."

2.2 Graph-to-Text Generation

In the aforementioned scenarios, where other graph presentation approaches prove ineffective or inappropriate, providing text descriptions of graphs is an appealing alternative. This approach gives rise to the graph-to-text generation problem, which entails automatically converting the information stored within a graph into a comprehensible and accurate natural language description. This task bridges the gap between structured data representation and human-readable content and thus facilitates the synthesis of informative and contextually relevant textual narratives from graph-based data sources.

The process of graph-to-text generation involves several key challenges. First, the system must effectively understand and navigate the graph's structure, identifying the most relevant and significant information to include in the text. Second, it needs to generate text that not only accurately represents the data but also maintains naturalness, coherence, and fluency, adhering to the conventions of human language. This requires sophisticated linguistic and contextual understanding, as well as the ability to generate text that aligns with the intended message and purpose.

Approaches to graph-to-text generation often involve advanced natural language processing (NLP) methodologies, including deep learning models such as Transformers [Vaswani et al., 2017], which can handle the complexity and non-linear nature of graph data. PLM models [Raffel et al., 2020, Lewis et al., 2020, Touvron et al., 2023] were trained on large text datasets to learn the mapping between input sequences and output sequences, often employing attention mechanisms to focus on different parts of the input sequences when generating corresponding text segments.

While language models enable graph-to-text generation, in turn graph-to-text generation enriches language models. Converting knowledge graph into natural language text facilitates its seamless integration into existing language models. This

transformation allows for the sophisticated blending of structured data with unstructured text, enhancing the linguistic capabilities of these models. According to Agarwal et al. [2021], verbalizing a knowledge graph for integration with natural text corpora can be used in language models' pre-training, which significantly improves PLMs' performance in open-domain question-answering systems. This enhancement underscores the value of incorporating structured knowledge into the broader framework of AI language processing.

2.3 Transformer

The Transformer model [Vaswani et al., 2017] represents a groundbreaking shift in the field of NLP. Prior to the advent of the Transformer, most state-of-the-art NLP models back then were based on recurrent neural networks (RNNs) [Schuster and Paliwal, 1997] and long short-term memory network (LSTMs) [Sundermeyer et al., 2012], which process data sequentially. This sequential processing posed significant limitations in terms of computational efficiency and the ability to handle long-range dependencies within the input.

The Transformer model overcomes these limitations by utilizing a mechanism known as *self-attention* that processes the entire input data simultaneously, which supports parallel computation and thus leads to significantly improved efficiency. This parallelism not only accelerates training but also enables the model to effectively capture long-range dependencies in the data. The Transformer's architecture, devoid of recurrence, relies entirely on attention mechanisms to draw global dependencies between input and output, making it uniquely suited for handling a wide range of sequence-to-sequence tasks, such as machine translation [Wang et al., 2019], text summarization [Liu and Lapata, 2019], and more.

Encoder and decoder. The Transformer's architecture is composed of two main parts: the encoder and the decoder. Each of these two parts contains multiple layers that consist of self-attention mechanisms and position-wise feed-forward networks. The encoder maps an input sequence to a sequence of tensor representations, which the decoder then uses to generate an output sequence. Both the encoder and decoder are made up of a stack of identical layers, each containing two sub-layers: a multi-head self-attention mechanism and a position-wise fully connected feed-forward network.

Self-attention mechanism. At the heart of the Transformer is the self-attention mechanism, which allows the model to weigh the significance of different parts of the input sequence when processing each word. For each word in the input sequence, the self-attention mechanism computes a score that reflects its relevance to other words. These scores are used to compute a weighted sum of the values, which represents the output of the attention layer. Mathematically, this is represented as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.1}$$

where Q, K, and V are the query, key, and value matrices, respectively, and d_k is the dimension of the key vectors.

Positional encoding. Since the Transformer does not inherently capture the sequential nature of the data, positional encoding is added to the input embeddings at the bottom of the encoder and decoder stacks. This encoding provides the model with information about the position of the words in the sequence, ensuring that the sequence order affects the computations of the model.

2.4 Pre-training and Fine-tuning

Pre-training and fine-tuning are two crucial stages in the development of machine learning models, particularly in the field of natural language processing.

Pre-training involves training a model on a large, diverse dataset before it is used for a specific task. This stage is designed to help the model learn a broad understanding of language or other input features from extensive, generalized data sources. The objective is to capture a wide range of linguistic patterns, relationships, and structures that are common across various texts or scenarios. Models such as BERT (Bidirectional Encoder Representations from Transformers) [Devlin et al., 2019], GPT (Generative Pre-trained Transformer) [Radford et al., 2018], and other Transformer-based architectures typically undergo pre-training on vast corpora like books, websites, and other text-rich materials. Pre-training techniques leverage various strategies to enable models to learn general patterns from large datasets before being fine-tuned for specific tasks. Some of the most common pre-training techniques are as follows.

- **Masked language modeling (MLM) [Salazar et al., 2020]**. Used in models such as BERT, this technique randomly masks words in the input sentences and trains the model to predict the masked words based only on their context. This encourages the model to develop a deep understanding of language context and relationships between words.

- **Autoregressive language modeling [Yang et al., 2019]**. This approach, used in models such as GPT, trains the model to predict the next word in a sentence given the words that precede it. This sequential prediction task helps the model to learn both syntax and long-range dependencies within text.

- **Autoencoding [Kingma and Welling, 2014]**. Autoencoding is an unsupervised learning technique to learn a compressed, distributed representation (encod-

ing) of input data, and then reconstruct the data from this encoding with minimal loss. The aforementioned MLM objective can be seen as a form of autoencoding where parts of the input are masked (treated as noise) and the model learns to reconstruct the original text.

- **Next sentence prediction [Devlin et al., 2019]**. This technique, originally used in the pre-training of BERT, involves presenting the model with two sentences and training it to predict whether the second sentence is a logical follow-up to the first. This helps in understanding relationships between sentences and is useful for tasks that involve reasoning over multiple sentences.

- **Contrastive learning [Chuang et al., 2020]**. This approach is used to train models to distinguish between similar and dissimilar items. In NLP, this might involve contrasting sentences or documents to enhance the model's understanding of nuanced differences in text.

- **Permutation language modeling [Yang et al., 2019]**. Utilized by models such as XLNet [Yang et al., 2019], this technique permutes the order of words in the input data and trains the model to predict the original order of words. It combines the strengths of both autoregressive and autoencoding techniques.

- **Electra-style training [Clark et al., 2020]**. Instead of only predicting masked words, this method trains a discriminator model to distinguish between "real" and "fake" input tokens generated by a separate generator model. This can be more efficient than traditional MLM because the model learns from every token in the input rather than just from masked ones.

- **Multi-task learning [Sener and Koltun, 2018]**. In this approach, the model is simultaneously trained on multiple different tasks during pre-training. This can include a mix of different types of language tasks (e.g., sentiment analysis, named entity recognition, syntactic parsing) to foster a broader understanding of language.

These techniques leverage large-scale datasets and computational power to create models that have a deep, generalized understanding of language patterns, which can then be adapted to specific tasks through fine-tuning [Howard and Ruder, 2018]. Fine-tuning follows pre-training and is a more targeted training process. In this stage, the pre-trained model is adapted to a specific task or dataset. Fine-tuning adjusts the weights and parameters of the pre-trained model so it can perform well on particular tasks such as sentiment analysis [Zhang et al., 2020], question answering [Su et al., 2019], or document classification [Pappagari et al., 2019]. This step is crucial because it allows the model to specialize its broadly acquired knowledge to the nuances and specific requirements of the task at hand. The fine-tuning process usually requires a smaller, task-specific dataset and fewer training cycles compared to pre-training, due to the foundational knowledge the model has already acquired.

Together, pre-training and fine-tuning enable the development of highly effective and adaptable models that combine the strengths of extensive learning with specialized task performance. Fine-tuning PLMs have achieved state-of-the-art performance on most NLP tasks [Min et al., 2023], including graph-to-text tasks [Ribeiro et al., 2021].

CHAPTER 3

LIMITATIONS OF EXISTING DATASETS

First, most previous models were trained on small handcrafted 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 handcrafted approach cannot scale to such as massive knowledge graphs, 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 handcrafted datasets such as WebNLG tend to follow monotonous templates, plausibly because the examples were written by a small number of human contributors. This limits the capability of trained models to use diverse expressions in narrating graph fragments. This lack of linguistic variation can hamper the 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 more general acyclic graphs (i.e., one or more trees). Figure 3.1 displays the distinct graph shapes in the WebNLG dataset, in descending order by number of instances, and Table 3.1 shows the corresponding instances numbers. The graphs in WebNLG have 41 distinct topological shapes, out of which 32 are acyclic graphs. The cycles are all 2-edge loops or self-loops. In DART, 83% of the graphs are star 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 30% acyclic instances and only 2% star graphs. However 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 Chapter 4, Figure 4.6) show that only 2 of the 10 most frequent graph formations [1] are star graphs, and 3 are cyclic graphs.

rank	**#instances**	**rank**	**#instances**	**rank**	**#instances**	**rank**	**#instances**
1	5965	11	276	21	28	31	12
2	4759	12	231	22	27	32	11
3	3799	13	231	23	25	33	9
4	2648	14	228	24	23	34	8
5	1421	15	154	25	19	35	6
6	1365	16	139	26	17	36	6
7	1332	17	127	27	14	37	6
8	832	18	87	28	14	38	5
9	695	19	76	29	13	39	3
10	625	20	46	30	12	40	3
						41	1

Table 3.1: WebNLG graph shapes and the corresponding number of instances

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

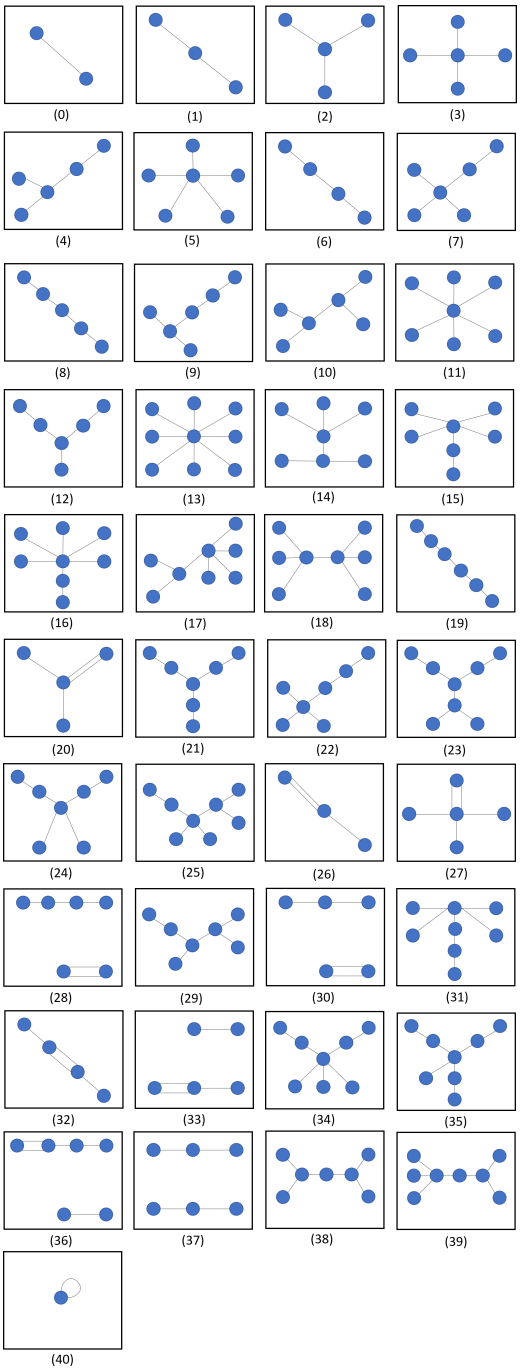


Figure 3.1: WebNLG graph shapes

CHAPTER 4

THE GraphNarrative DATASET

This chapter explains how we generated our new dataset GraphNarrative by aligning Wikipedia texts with Freebase. Note that the methodology could be applicable to text corpora beyond Wikipedia and knowledge graphs beyond Freebase. This section also contrasts GraphNarrative with existing benchmark datasets to demonstrate how it addresses current datasets' limitations.

4.1 Dataset Creation

4.1.1 Pre-processing of Text Corpus and Knowledge Graph

We used the Wikipedia dump released on Sep. 1st 2019 as our text corpus. [1] The raw dump was in the form of wikitext source and metadata embedded in XML. To preprocess the raw dump, we utilized WikiExtractor, an existing tool. [2] The tool transformed the compressed XML file into numerous plain text files containing bodies of Wikipedia articles without tables, infoboxes, table of contents, categories, and so on.[3]

Our Freebase knowledge graph is from [Shirvani-Mahdavi et al., 2023] which used the most recent Freebase dump [4] as the data source. 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

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

[2] https://github.com/attardi/wikiextractor

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

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

edge out of each redundant pair. Hence, we did exactly that in pre-processing the whole Freebase dump so that our input graphs have no reverse edges. Furthermore, our pre-processing also removed the mediator (CVT) nodes [Bollacker et al., 2008] by concatenating edges connected through mediator nodes. Below we explain these pre-processing steps in more detail, which was also discussed in [Shirvani-Mahdavi et al., 2023].

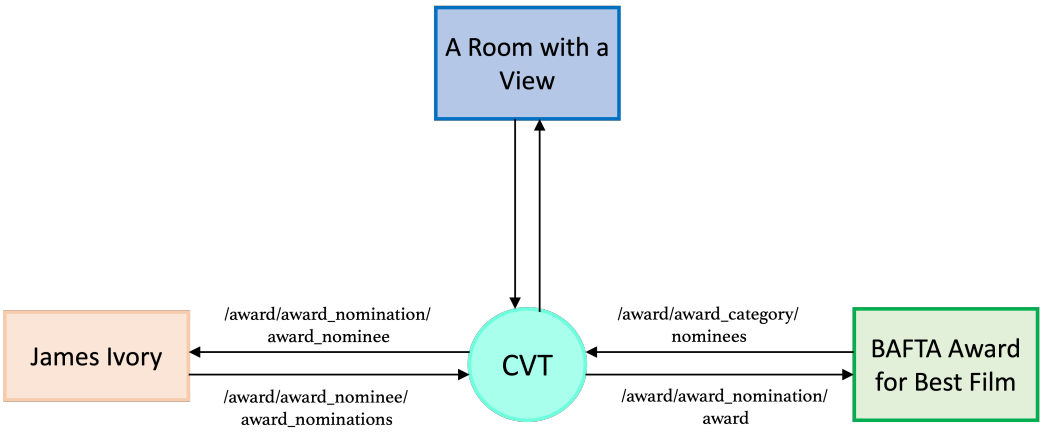
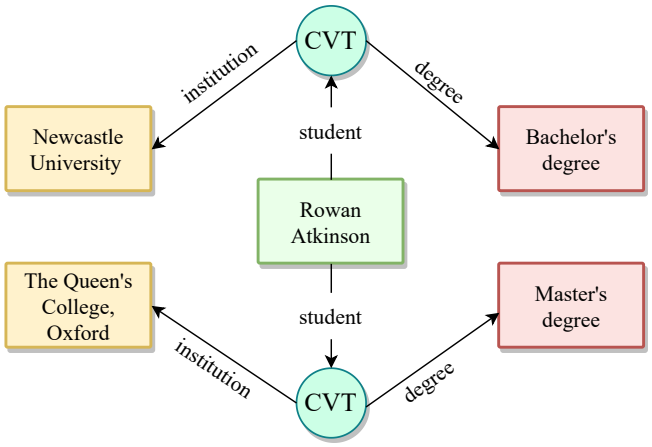
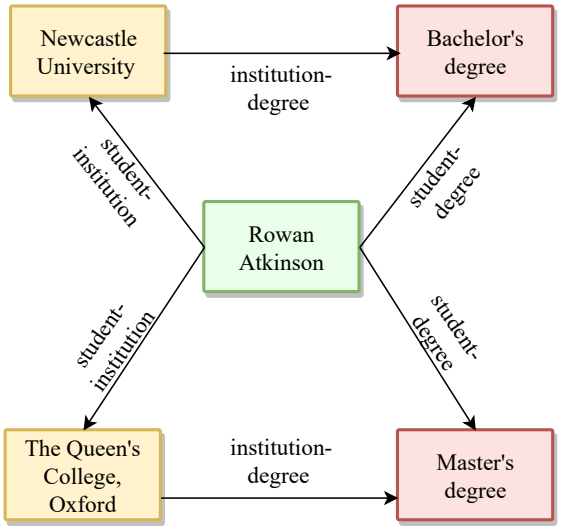


Figure 4.1: A small fragment of Freebase, with an intermediator node

Reverse triples. When a new fact was added to Freebase, it was represented as a pair of reverse triples (`s`, p, `o`) and (`s`, p^{-1}, `o`), where p^{-1} is the inverse of p. Freebase explicitly denotes reverse relations using a special relation */type/property/reverse_property* [Pellissier Tanon et al., 2016, Färber, 2017]. For example, */film/film/directed_by* and */film/director/film* are reverse relations, as indicated by triple (`/film/film/directed_by`, */type/property/reverse_property*, `/film/director/film`). Consequently, (`A Room With A View`, */film/film/directed_by*, `James Ivory`) and (`James Ivory`, *film/director/film*, `A Room With A View`) form reverse triples, depicted as two edges in opposite directions in Figure 4.1. A triple (`r1`, */type/property/reverse_property*, `r2`) signifies that relations *r1* and *r2* are inverses of each other. These reverse rela-



(a) Before transformation: n-ary relationships modeled via CVT nodes



(b) After transformation: binary relationships without CVT nodes

Figure 4.2: An example of information loss after converting n-ary relationships to binary relationships

tionships are semantically redundant; hence, we eliminate the triples involving one relationship of the reverse pair. When removing reverse triples, i.e., triples that belong to reverse relations, we discard all triples associated with relation *r2*.

Intermediate nodes. *Intermediate nodes*, also known as mediator nodes and CVT nodes, are utilized in Freebase to represent n-ary relationships [Pellissier Tanon et al., 2016]. For instance, Figure 4.1 illustrates a CVT node connected to an `award`,

a `nominee`, and a `work`. This or a similar approach is necessary for accurate modeling of real-world data. However, this modeling complexity poses a significant challenge for tasks such as graph-to-text conversion. We acknowledge that the creation of our dataset does not account for multiary relationships in knowledge graphs. Specifically, the Freebase version used in our work has multiary relationships converted into binary relationships [Shirvani-Mahdavi et al., 2023]. Generally, there is a lack of research into multiary relationships in graph-to-text models. To the best of our knowledge, the only work in this area that addresses multiary relationships is [Agarwal et al., 2021], and they also converted multiary relationships into binary ones. We also acknowledge this limitation in Chapter 9.

For better clarity, we use Figure 4.2 to further illustrate the conversion from multiary relationships into binary relationships. The edge (relationship) labels in Freebase are structured as /[domain]/[type]/[label]. The /[domain]/[type] prefix identifies the subject entity's type that an edge belongs to, while [label] provides an intuitive meaning of the relationship. For simplicity of presentation, we use [label] instead of /[domain]/[type]/[label] to denote edge labels. In the post-transformation graph, the label of a concatenated edge is the concatenation of the two original edge labels. Figure 4.2a depicts two ternary relationships about `Rowan Atkinson`'s educational institutions and his degrees. Figure 4.2b is after transforming the ternary relationships into binary relationships. Note that, one may convert an n-ary relationship centered at a CVT node into $\binom{n}{2}$ binary relationships between every pair of entities, by concatenating the edges that connect the entities through the CVT node. Note that the transformation could lead to loss of information [Wen et al., 2016] and is irreversible [Rossi et al., 2021]. More details regarding the information loss can be found in [Shirvani-Mahdavi et al., 2023].

4.1.2 Graph-Text Alignment

It is impractical to rely on manual efforts to write sentences describing subgraphs and form a dataset of proper size because of the large volume and diversity of entities and relationships in large-scale open knowledge graphs. Our approach is to automatically generate training examples by aligning Wikipedia sentences with corresponding Freebase subgraphs. 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. See Figure 4.3 for an example. This is achieved by an *entity linking* step followed by an *edge detection* step.

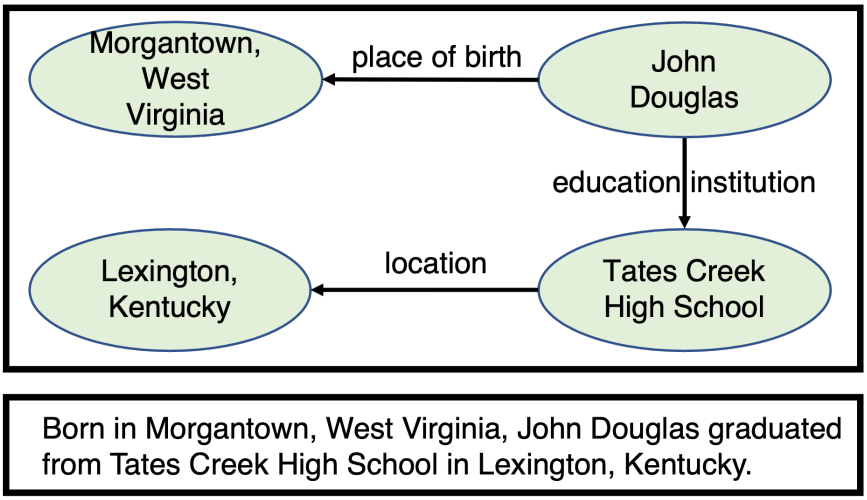


Figure 4.3: A graph-sentence pair in GraphNarrative

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.

For *coreference resolution*, we applied the implementation [Lee et al., 2017] in AllenNLP [Gardner et al., 2018] with default settings on Wikipedia articles to replace

the token spans of pronouns and aliases with the corresponding entities they refer to. This helps produce more graph-text pairs for GraphNarrative. We conducted human evaluation to assess the quality of the coreference resolution results on 20 randomly selected Wikipedia articles. The assessment yielded a precision of 91.4% (630 of the 689 resolved entity coreferences were correct) and a recall of 98.3% (11 entity coreferences were missed).

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.

The *Wikipedia-to-Freebase entity mapping* created 4,408,115 one-to-one mappings between English Wikipedia entities (i.e., articles) and their corresponding Freebase entities, through a combination of three engineering methods—by using existing mapping in Freebase, by using Wikidata as the midpoint connecting Wikipedia and Freebase entities, and similarly by using DBpedia [Auer et al., 2007] as the midpoint. More specifically, the mapping was created by employing the following three methods. 1) Parsing the Freebase data dump to obtain a Wikipedia-to-Freebase entity mapping using `https://github.com/saleiro/Freebase-to-Wikipedia`. 2) Inferring from a Wikipedia-to-Wikidata mapping in wikimapper [Klie, 2022] and a Wikidata-to-Freebase mapping at `https://developers.google.com/freebase`. 3) Inferring from the Wikipedia-to-DBpedia and the DBpedia-to-Freebase mappings at `http://downloads.dbpedia.org/2016-10/core-i18n/en/`. The overall Wikipedia-to-Freebase entity mapping is obtained by combining all three methods and eliminating conflicting entity mappings. The mapping file link can be found at `https://github.com/idirlab/graphnarrator`.

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 relationship 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 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 (detailed assessment in Section 7.2). Nevertheless, our workflow of dataset creation allows for more advanced and accurate methods in each component.

4.2 Characteristics of GraphNarrative

This section qualitatively and quantitatively analyzes how GraphNarrative bridges the gap between graph-to-text models and real-world settings.

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. Table 4.1 shows the number of instances in GraphNarrative in each Freebase domain. Table 4.2 compares five datasets: WebNLG, AGENDA, EventNarrative, TEKGEN, and GraphNarrative. The datasets differ in terms of their knowledge graph source, text corpus, domain coverage, number of instances, number of entities, number of triples, number of relations, and percentage of star graphs. Our dataset, which uses Freebase as the knowledge graph and Wikipedia as the text

Domain	Count	Domain	Count	Domain	Count	Domain	Count
location	2,291,039	people	1,557,482	sports	633,854	music	631,159
government	432,881	organization	420,824	education	405,418	film	386,237
tv	264,398	book	218,885	award	166,832	military	146,581
soccer	122,931	time	88,337	geography	87,935	business	85,587
olympics	80,622	transport-ation	60,669	broadcast	56,669	baseball	56,343
fictional universe	37,185	biology	37,121	influence	36,370	language	32,796
computer	31,059	cvg	30,873	aviation	29,992	architect-ure	29,092
protected sites	27,720	religion	27,158	symbols	26,881	travel	25,526
visual art	19,358	basketball	19,259	royalty	17,960	astronomy	17,847
american football	15,067	metropoli-tan transit	14,585	comic books	11,312	law	10,095
media common	9,538	spaceflight	7,441	tennis	6,553	boats	6,550
medicine	5,317	meteorology	5,084	automotive	4,857	theater	4,816
internet	3,478	amusement parks	3,027	event	2,562	comic strips	2,532
games	1,978	food	1,613	martial arts	1,448	measure-ment unit	1,289
ice hockey	1,195	finance	1,184	rail	1,175	opera	1,157
cricket	1,027	dining	706	projects	600	zoos	402
venture capital	389	digicams	220	distilled spirits	213	skiing	197
conferences	178	exhibitions	139	comedy	137	boxing	124
geology	87	engineering	78	chemistry	74	library	70
chess	65	fashion	59	interests	45	celebrities	42
wine	22	radio	13	bicycles	13	physics	1

Table 4.1: Number of GraphNarrative instances in each Freebase domain

corpus, stands out by covering the open domain with a large number of instances, entities, triples, and relations. Most other datasets are significantly smaller in these aspects.

Linguistic variation. Using Wikipedia as the corpus, the graph-text pairs in GraphNarrative allow a model to learn from many Wikipedia authors' diverse nar-

	WebNLG	**DART**	**AGENDA**	**Event Narrative**	**TEKGEN**	GraphNarrative
Knowledge Graph	DBpedia	N/A	N/A	Wikidata	Wikidata	Freebase
Text Corpus	Handcraft	Handcraft	Scientific abstract	Wikipedia	Wikipedia	Wikipedia
Domain	15 DBpedia categories	N/A	Scientific research	Events	Open domain	Open domain
Instances	25,298	38,391	40,720	224,428	7,895,789	**8,769,634**
Entities	2,730	27,000	159,691	305,685	**4,856,439**	1,853,752
Triples	3,221	32,139	177,568	649,337	11,373,838	**15,472,249**
Relations	354	**3,834**	7	672	663	1,724
Star Graphs	57%	83%	2%	94%	96%	22%

Table 4.2: Comparison of characteristics of the datasets used for graph-to-text generation

rations. On the contrary, text in handcrafted datasets such as WebNLG and DART tend to follow monotonous templates from a small number of human contributors.

Graph structure complexity. The graphs in GraphNarrative contain 1–15 triples and 2–20 entities, in 7,920 distinct topological shapes based on graph isomorphism. The distributions of graphs in GraphNarrative by numbers of triples and entities are shown in base-10 logarithmic scale in Figure 4.4 and Figure 4.5, respectively. Furthermore, the distribution of distinct GraphNarrative graph shapes by number of entities is in Table 4.3. Figure 4.6 displays the 10 most frequent shapes along with their instance counts. It is not limited to star graphs, but also includes triangles, paths, etc. In fact, only 22% of the instance graphs are star graphs. On the contrary, EventNarrative and TEKGEN are dominated by star graphs, as Table 4.2 shows.

#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	all
#shapes	542	378	222	176	106	58	52	22	12	7920

Table 4.3: Distribution of distinct GraphNarrative graph shapes by number of entities

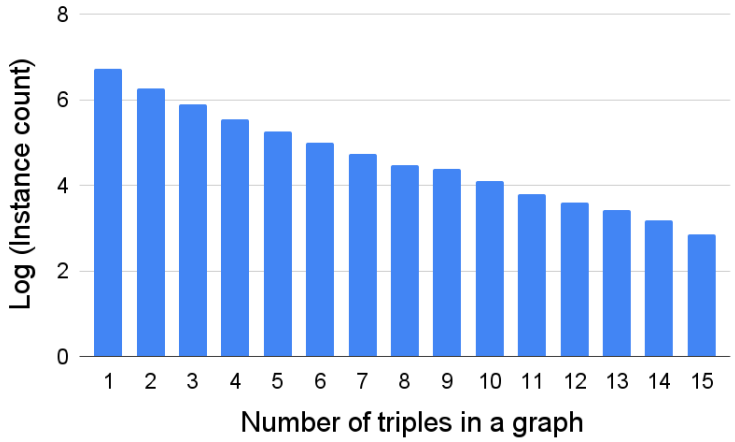


Figure 4.4: Distribution of GraphNarrative instances by number of triples in graphs

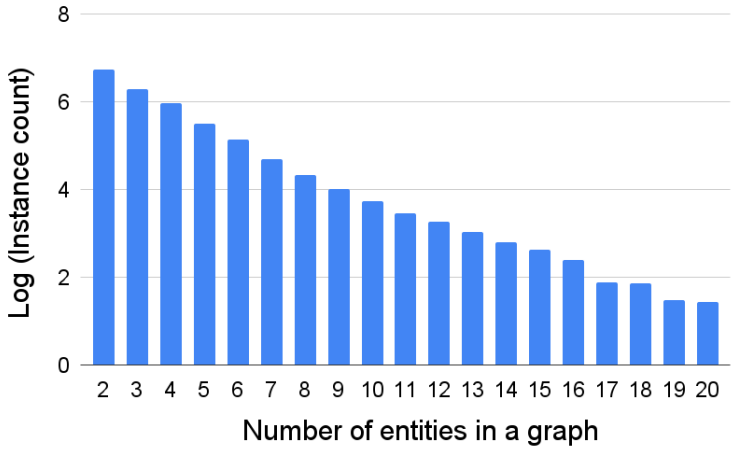


Figure 4.5: Distribution of GraphNarrative instances by number of entities in graphs

Text distribution. The dataset exhibits an average sentence length of 34.68 tokens for original sentences and 20.66 tokens for trimmed sentences (sentences obtained by applying trimming (Chapter 6) on the original sentence). Table 4.4 provides detailed distribution of sentence lengths. Table 4.5 presents the average sentence token counts by number of triples in the graphs. It underscores that our model was trained using a diverse set of examples, including those with lengthy sentences and a substantial number of triples.

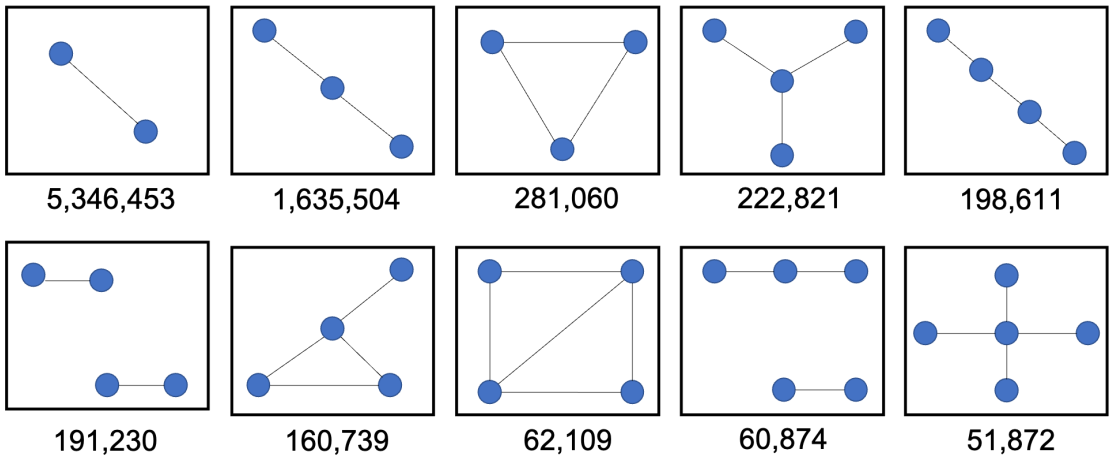


Figure 4.6: The 10 most frequent graph shapes in GraphNarrative, with instance counts

#Tokens	Original Sentence	Trimmed Sentence
0-10	64,861 (0.74%)	1,734,759 (19.78%)
10-20	1,442,582 (16.45%)	3,369,747 (38.43%)
20-30	2,592,651 (29.56%)	1,944,225 (22.17%)
30-40	2,048,174 (23.36%)	919,554 (10.49%)
40-50	1,215,928 (13.87%)	416,187 (4.75%)
50-60	649,403 (7.41%)	190,363 (2.17%)
60-70	339,671 (3.87%)	91,289 (1.04%)
70-80	181,699 (2.07%)	45,38 (0.52%)
80-90	97,618 (1.11%)	23,823 (0.27%)
90-100	53,981 (0.62%)	13,299 (0.15%)
100-110	83,066 (0.95%)	21,008 (0.24%)

Table 4.4: Distribution of GraphNarrative instances by sentence length

#Triples	#Tokens	#Triples	#Tokens	#Triples	#Tokens
1	14.15	2	20.32	3	23.12
4	26.29	5	28.21	6	29.74
7	31.12	8	33.41	9	30.39
10	35.90	11	39.53	12	42.49
13	43.60	14	49.71	15	50.21

Table 4.5: Average GraphNarrative sentence length by number of triples in graphs

CHAPTER 5

MODELS

A knowledge graph, denoted as $\mathcal{G}$, is composed of a set of triples (s, p, o), where the subject s and object o are entities from the entity space E, and predicate p are from the relation space R. The task of generating text from a subgraph $G \subset \mathcal{G}$ can be formulated as, given G, modeling the probability distribution of n-token sequences $Y = (y_1, ..., y_n)$ that describe G.

Existing graph-to-text models often use a decoder-only [Brown et al., 2020] or encoder-decoder structure [Sutskever et al., 2014], where the encoder learns representations of input graphs and the decoder subsequently translates the representations into token sequences. In decoder-only models, e.g., GPT-2 [Radford et al., 2019], GPT-3 [Brown et al., 2020], ChatGPT (i.e., GPT-3.5) [OpenAI, 2022], and GPT-4 [Achiam et al., 2023] in the GPT family, the decoder uses text sequence embedding as the representation of input graphs. Encoder-decoder models fall into two main categories based on graph representations—1) sequence-to-sequence models that encode linearized graphs' token sequences with LSTMs [Trisedya et al., 2018, Gardent et al., 2017a] or Transformers [Castro Ferreira et al., 2019], and 2) models that use a dedicated graph encoder to capture the structural information of knowledge graphs [Schmitt et al., 2021, Ribeiro et al., 2020, Marcheggiani and Perez-Beltrachini, 2018].

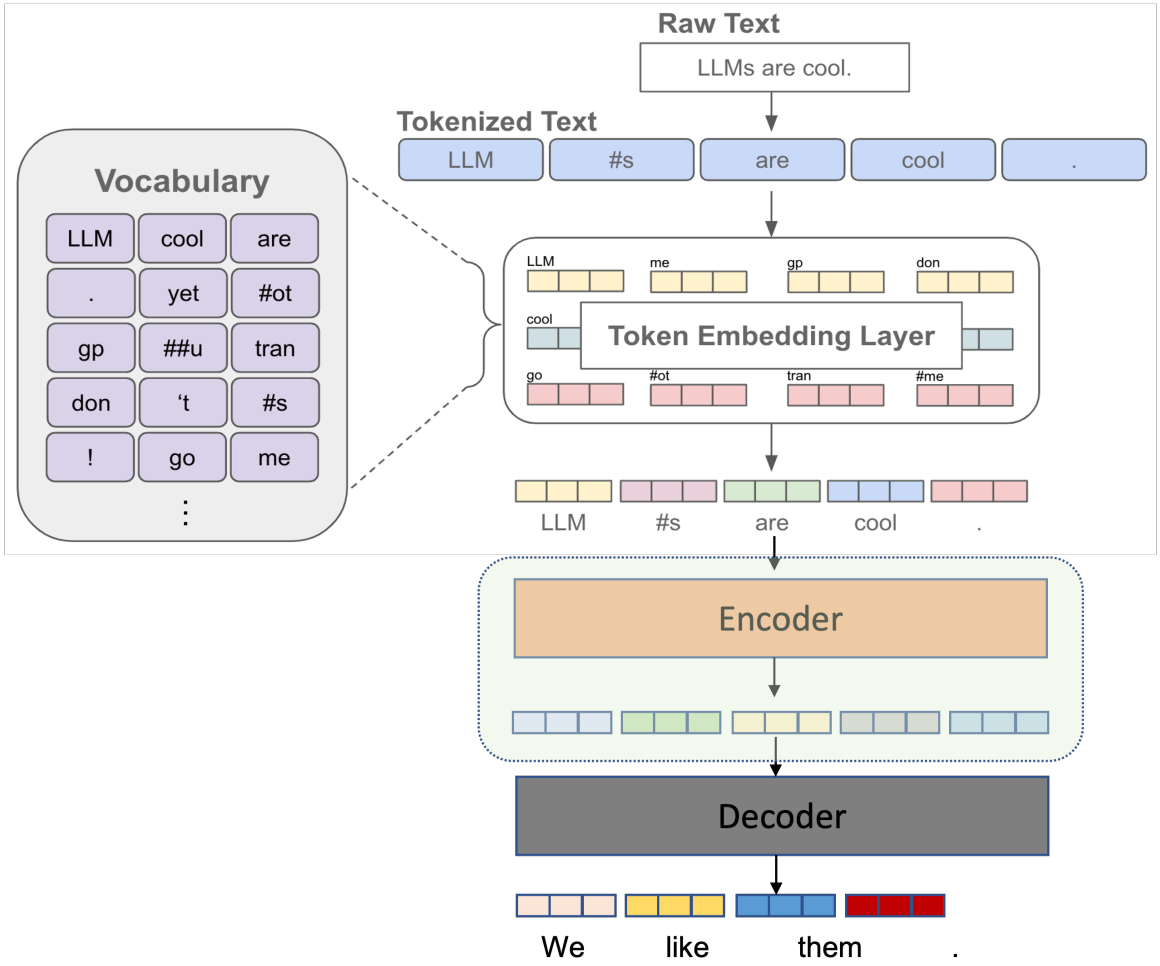


Figure 5.1: Transformer-based pre-trained language models

5.1 Transformer-based Pre-trained Langauge Models

Figure 5.1 illustrates the basic steps of how Transformer-based pre-trained language models operate. The model takes a sentence as input and tokenizes it into word tokens. For each token, a vocabulary lookup maps it to a vector. These vectors are then input to the encoder, which learns a hidden representation using the self-attention mechanism and positional embeddings. The hidden representation is subsequently fed into the decoder layer. The decoder predicts the probability of each token in the vocabulary as the next token, based on the current input, and outputs

the one with the highest probability. This process continues until a special end token is predicted or the maximum sequence length is reached. The aforementioned decoder-only models do not have an encoder layer and use the embedding lookup table directly as the representation of the input sequences.

Model	**BART (Bidirectional Auto-Regressive Transformers)**	**T5 (Text-to-Text Transfer Transformer)**	**LLaMA (Large Language Model Meta AI)**
Structure	Transformer (encoder + decoder)	Transformer (encoder + decoder)	Transformer (decoder only)
Pre-training Corpus	Wikipedia, news articles, books, and subsets of Common Crawl dataset (160GB text)	Books, Wikipedia, and the Common Crawl dataset (750 GB text)	CommonCrawl, C4, Github, Wikipedia, Books, ArXiv, StackExchange
Pre-training Strategy	Self-supervised: recover corrupted sentences	Self-supervised: recover corrupted sentences, Supervised: frame many tasks as sequence-to-sequence ones	Self-supervised: recover corrupted sentences
Number of Model Parameters	base(140M), large(400M), ...	small(60M), base(220M), large(770M), ...	TinyLLaMA(1.1B), LLaMA-2(7–70B), LLaMA-3(8–70B), ...

Table 5.1: Comparison of BART, T5, and LLaMA models

Table 5.1 provides a brief comparative summary of three advanced language models—BART, T5, and LLaMA [Touvron et al., 2023]. Each model utilizes the Transformer architecture, with BART and T5 incorporating both an encoder and a decoder, while LLaMA uses only a decoder.

In terms of the pre-training corpus, BART uses a combination of Wikipedia, [1] news articles, books, and parts of the Common Crawl dataset, [2] totaling 160GB of text. T5, on the other hand, has been trained on a larger corpus including books, Wikipedia, and the full Common Crawl dataset, amounting to 750GB. LLaMA has

[1] `https://www.wikipedia.org/`

[2] `https://commoncrawl.org/`

been trained on an even more extensive range of sources, including Common Crawl, C4, [3] GitHub, [4] Wikipedia, books, ArXiv, [5] and StackExchange. [6]

For pre-training strategies, BART and LLaMA employ a self-supervised approach that focuses on recovering corrupted sentences. T5 also uses a self-supervised approach but adds a supervised component, treating many tasks as sequence-to-sequence problems.

The models differ significantly in terms of number of parameters: BART has versions ranging from 140M to 400M parameters; T5 has models from 60M up to 770M parameters; and LLaMA starts with the TinyLLaMA at 1.1 billion parameters, with other larger versions scaling between 7 billion and 70 billion parameters.

• **T5**. The T5, or Text-to-Text Transfer Transformer, simplifies the NLP pipeline by treating both input and output as text strings, allowing for consistent application across diverse tasks such as translation, summarization, question answering, and classification. Built on the original Transformer architecture, T5 introduces several innovations, notably in its pre-training process. It uses a "span-corruption" objective, where random spans of text are masked and the model is trained to predict these spans, enhancing context understanding and text generation. T5 also employs a relative position encoding scheme, which replaces absolute position encodings, to better capture word relationships and improve handling of long-range dependencies.

• **BART**. The BART (Bidirectional and Auto-Regressive Transformer) model merges the strengths of bidirectional Transformers (e.g., BERT) and auto-regressive Transformers (e.g., GPT). BART's architecture includes a standard Transformer setup with an encoder and a decoder. The encoder is similar to BERT's, while

[3] `https://www.tensorflow.org/datasets/catalog/c4`

[4] `https://github.com/`

[5] `https://arxiv.org/`

[6] `https://stackexchange.com/`

the decoder is akin to GPT's. This makes BART versatile and proficient in handling both generation and understanding tasks. A key feature of BART is its pre-training method, which uses a noise-aware denoising autoencoder framework. The model is trained on systematically corrupted text and learns to reconstruct the original text.

• **LLaMA**. The LLaMA (Large Language Model from Meta AI) delivers state-of-the-art performance across various NLP tasks. Designed to be more accessible and efficient than predecessors such as GPT-3, as well as larger BART and T5 models, LLaMA stands out due to its innovative approaches to model training and scaling. Key features include model and data parallelism, and the use of efficient training methodologies such as mixed-precision training and gradient checkpointing. These techniques not only enhance its power and size but also make it highly cost-effective [Touvron et al., 2023].

5.2 Models Used

Fine-tuning Transformer-based, sequence-to-sequence encoder-decoder PLMs (e.g., T5 and BART) achieved state-of-the-art performance on WebNLG [Ribeiro et al., 2021, Wang et al., 2021, Clive et al., 2021] and DART [Aghajanyan et al., 2021]. Yuan and Färber [2023] reported that fine-tuning T5 and BART on WebNLG and AGENDA datasets yielded better results than zero-shot learning using GPT-3 and GPT-3.5. They also reported factual hallucinations from GPT-3 and GPT-3.5. While the reported results from fine-tuning GPT-2 on WebNLG [Harkous et al., 2020] are worse than the current state-of-the-art, no results have been reported based on fine-tuning GPT-3, GPT-3.5, or GPT-4.

Following the state-of-the-art approach, we also fine-tuned T5 (small, base, large), BART (base, large), and TinyLLaMA on GraphNarrative and other datasets in comparison.

5.3 Linearize the Graphs

PLMs are sequence-to-sequence models. We linearize graphs to convert them to token sequences which become input to PLMs. We explored two linearization methods, basic linearization and structured linearization. Figure 5.2 illustrates and contrasts these two methods. In both methods the token sequences incorporate special tokens to mark different positions within each triple—<H>, <R> and <T>, denoting subjects, relations and objects, respectively.

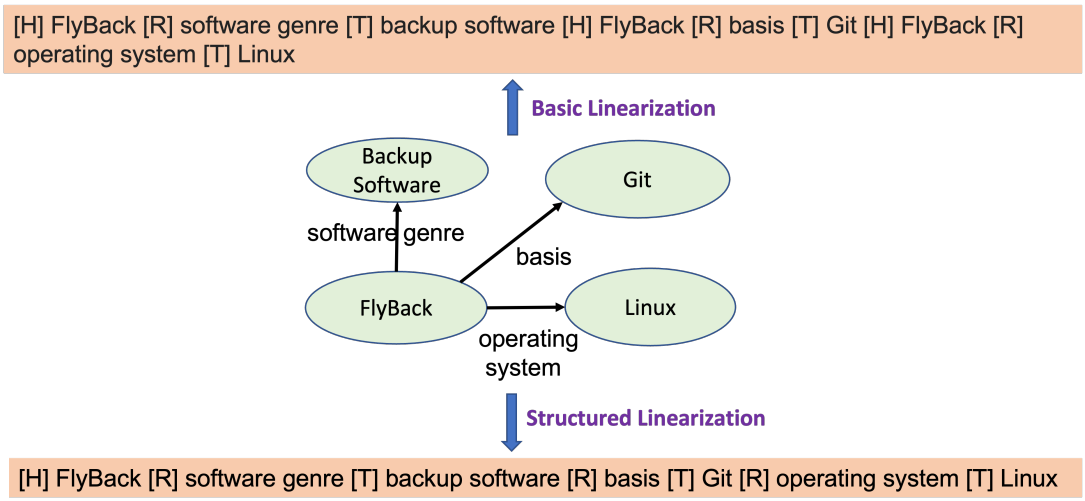


Figure 5.2: Two linearization methods

Basic Linearization. Basic linearization transforms a graph into a sequence of tokens by concatenating individual triples. Following the method in [Ribeiro et al., 2021], the graph in Figure 4.3 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".

Structured Linearization. The hierarchical and structural relationships between nodes (entities) in a graph, such as parent-child or ancestor-descendant re-

lationships, can be obscured when translated into a flat sequence. To maintain more structural integrity, we attempted structured linearization. Instead of simply concatenating triples in an unordered manner, we organized triples by grouping them based on common subjects. More specifically, the triples with the same subject are concatenated together, sharing one special token <H>. For instance, with structured linearization, the graph in Figure 4.3 is linearized as "<H> John Douglas <R> place of birth <T> Morgantown, West Virginia <R> education institution <T> Tates Creek High School <H> Tates Creek High School <R> location <T> Lexington, Kentucky". Triples (`John Douglas`, *place of birth*, `Morgantown, West Virginia`) and (`John Douglas`, *education institution*, `Tates Creek High School`) share the same subject `John Douglas` and they are concatenated together.

CHAPTER 6

MITIGATION OF HALLUCINATION

The culprit of the hallucination problem discussed in Chapter 1 is fabrication in training 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. Hallucinated facts are seldom found in the clean, manually-crafted WebNLG but are present in automatically extracted graph-text pairs in TEKGEN due to extraction errors.

6.1 Two Directions Toward Addressing Graph-to-Text Hallucination

There could be two plausible directions in tackling graph-to-text hallucination. One is to improve our graph-text alignment method (Section 4.1). The graph extracted from a piece of text during alignment may miss certain entities or relationships due to either extraction errors or disparities between the text corpus and the knowledge graph. The resulting graph-text pair may misguide the trained model to hallucinate facts. A more accurate alignment method can reduce such erroneous pairs and thereby reduce hallucination. However, this method has an inherent limitation—since a knowledge graph in real-world is often far from complete, there will be facts in text that cannot be mapped to the knowledge graph.

The other direction is to force the graph-text pairs to be consistent by trimming the sentences in graph-text pairs so as to keep only subsequences of the original sentences that correspond to the entities and relations depicted in the graphs.

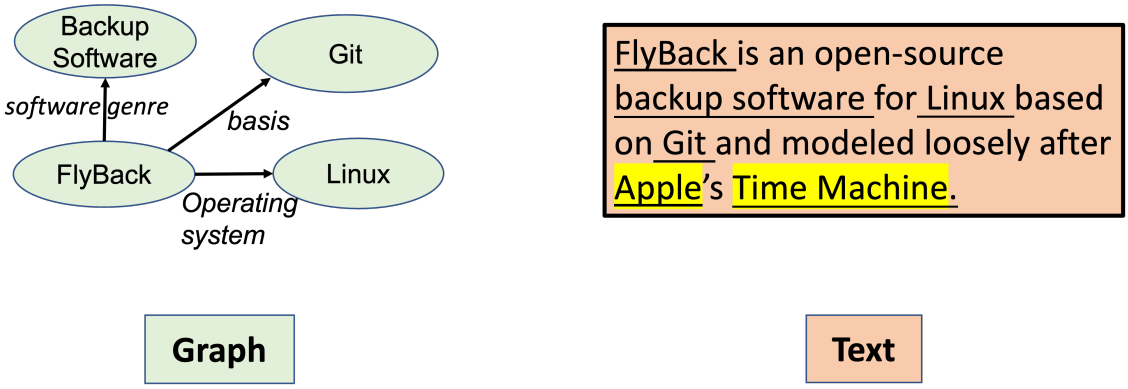
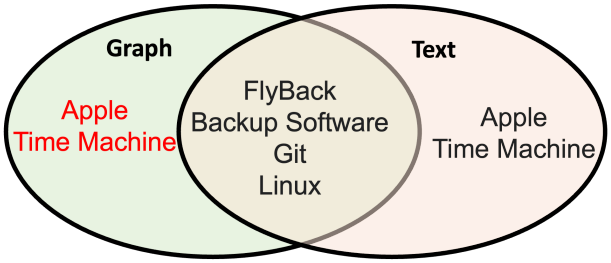
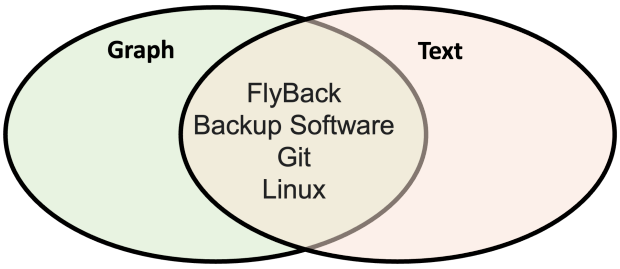


Figure 6.1: An example graph-text pair obtained from graph-text alignment



(a) Direction 1: develop more accurate graph-text alignment



(b) Direction 2: trim the sentences

Figure 6.2: Two directions of approaches to mitigating hallucination in graph-to-text

Figure 6.1 shows a graph-text pair obtained through the graph-text alignment method introduced in Section 4.1.2. Using this example, Figure 6.2 illustrates the two approaches to mitigating graph-to-text hallucination. In Figure 6.2, each subfigure is a Venn diagram. The oval labeled "Graph" represents the set of entities present in the graph, while the oval labeled "Text" represents the set of entities mentioned in the text. The intersection $(T \cap G)$ thus contains the entities that appear in both the

text and the graph, e.g., `FlyBack`, `Backup Software`, `Git`, and `Linux` for the graph-text pair in Figure 6.1. The text includes two extraneous entities, `Apple` and `Time Machine`, which are not aligned with the graph, i.e., the entities are in $T - G$. Given this graph-text pair, the first approach aims to develop better alignment techniques to find the corresponding entities `Apple` and `Time Machine` in the graph. This would lead to adding these entities to the graph, thereby enriching the graph with the additional entities present in the text. The second approach seeks to trim the sentence to exclude these extraneous entities. More specifically, it removes the token sequences related to `Apple` and `Time Machine` from the sentence.

Note that Figure 6.2 captures only the entities, but graph-text alignment also involves relationships (edges) between entities. If we use the first approach, we not only add missing entities from $T-G$ to the graph but also need to add corresponding edges to accurately reflect the relationships. For example, if "Apple" and "Time Machine" are added to the graph, we need to ensure that their relationships with other entities in the graph are also captured. If we use the trimming approach, we need to remove more than just the entities from $T - G$. We also need to remove other words associated with these entities to maintain grammatical correctness and coherence in the sentence. Besides, although not shown in this specific example, it is possible to have entities in the graph that do not appear in the text, i.e., $G - T$ may not be empty. This situation is much rarer as we are extracting a graph based on the text but it could happen nevertheless when wrong Freebase entities and relationships are aligned to the text.

Algorithm 1: Sentence Trimming

Input: W: A co-reference resolved Wikipedia sentence; G: The graph for W based on graph-text alignment

Output: W_{trim}: The trimmed text sequence

$M \leftarrow \{\}$

foreach $(s, p, o) \in G$ **do**

  $s' \leftarrow s.remove(special\_tokens)$

  $o' \leftarrow o.remove(special\_tokens)$

  $W \leftarrow W.replace((s, o), (s', o'))$

  $M[s'], M[o'], M'[s], M'[o] \leftarrow s, o, s', o'$

$W_{tree} \leftarrow W.dependency\_parsing()$

$min\_pos, max\_pos \leftarrow W.length, 0$

foreach $(s, p, o) \in G$ **do**

  $sdp \leftarrow shortest\_path(W_{tree}, M'[s], M'[o])$

  foreach $node \in sdp$ **do**

    $min\_pos \leftarrow \min(min\_pos, node.start)$

    $max\_pos \leftarrow \max(max\_pos, node.end)$

$W_{trim} \leftarrow W[min\_pos : max\_pos]$

foreach $k \in M.keys()$ **do**

  $W_{trim} \leftarrow W_{trim}.replace(k, M[k])$

return W_{trim}

6.2 Sentence Trimming Algorithm

This study explores the second aforementioned direction in mitigating hallucination. Nevertheless, in principle, a way to combine the two approaches is open for investigation.

Given a (Freebase subgraph G, Wikipedia sentence W) pair produced by alignment, we introduce a *sentence trimming* algorithm (pseudocode in Algorithm 1) to turn W into a trimmed sentence W_{trim} by eliminating portions that are not present in G while preserving the sentence's main idea.

First, the algorithm parses W and generates its dependency parse tree (DPT) W_{tree}, using spaCy [Honnibal et al., 2020]. 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}. [1] It then finds the leftmost position index *min_pos* in sentence W among all tokens on 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* to *max_pos*.

In Algorithm 1, Lines 1–6 are to ensure that an entity consisting of multiple tokens is tokenized into one single token, the mapping M is for recovering the entity's tokens from W in producing W_{trim} (Lines 15–16), and the mapping M' is for finding the processed s', o' given triple (s, p, o). Lines 9–14 find the leftmost position *min_pos* and the rightmost position *max_pos* from W by scanning each triple (s, p, o) in G and finding the tokens on the corresponding SDPs. The variable *node* denotes a token on the SDP in W_{tree} between entities s and o. *node.start* and *node.end* denote *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. Suppose s or o appear

[1] We used NetworkX [Hagberg et al., 2008] to find the shortest path between two tokens in a sentence.

multiple times in the original sentence as $s_1, s_2, \ldots, s_m$ and $o_1, o_2, \ldots, o_n$, where $s_1 = s_2 = \ldots = s_m = s$ and $o_1 = o_2 = \ldots = o_n = o$, and s_q and o_j $(1 \leq q \leq m, 1 \leq j \leq n)$ are in ascending order by their positions in the original sentence. When finding the SDP tokens between s and o (Line 10), we find the SDP tokens between s_1 and o_n if s_1 appears before o_1, or the SDP tokens between o_1 and s_m if o_1 appears before s_1.

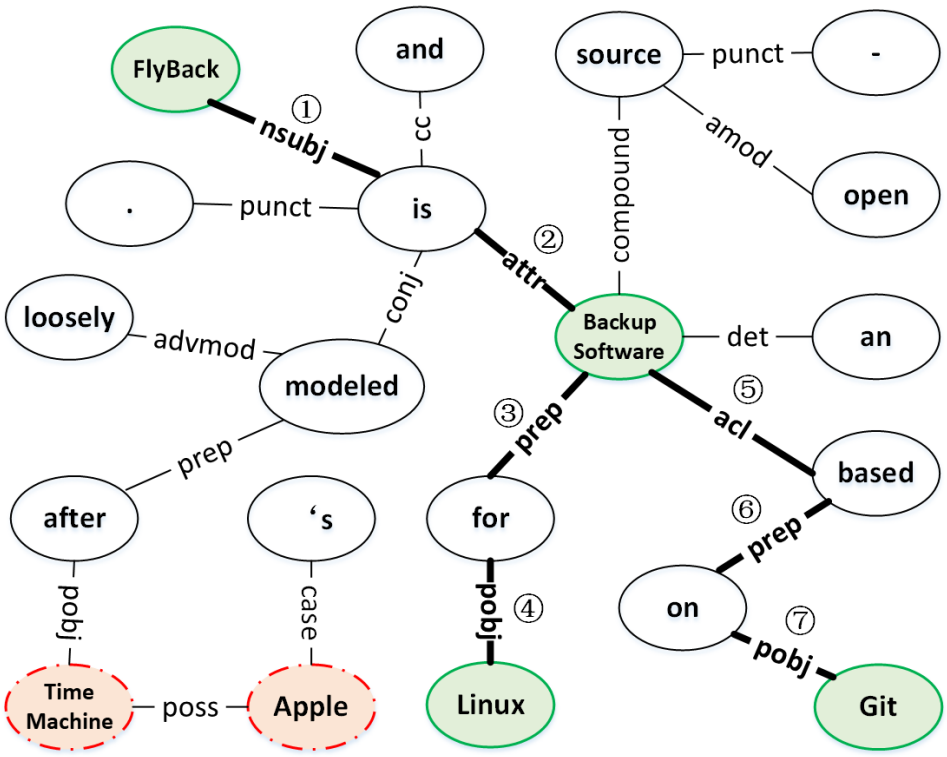


Figure 6.3: Dependency parse tree of the sentence "FlyBack is an open-source Backup Software for Linux based on Git and modeled loosely after Apple's Time Machine."

An example is in Figure 6.3 which 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. The circled numbers identify edges on the paths between subjects and objects. Given the 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 missing entities `Apple` and `Time Machine`, is trimmed from W.

Figures 6.4, 6.5, 6.6, and 6.7 illustrate the process of obtaining a trimmed sentence from a graph-text pair. The index numbers in square brackets following the tokens indicate their positions in the original sentence, which have different meanings than the circled numbers on the dependency parse tree in Figure 6.3. By iterating through each triple in the graph, the final trimmed sentence is produced. Given the example, the process is as follows. For the triple (`FlyBack`, *software genre*, `Backup Software`) in the graph, the SDP tokens on the DPT of the sentence are "FlyBack", "is" and "Backup Software". Their corresponding indices in the sentence are [1], [2] and [7], respectively. For the triple (`FlyBack`, *operating system*, `Linux`), the SDP tokens on the DPT of the sentence are "FlyBack", "is", "Backup Software", "for", and "Linux". And their corresponding indices in the sentence are [1], [2], [7], [8], and [9], respectively. For the triple (`FlyBack`, *basis*, `Git`), the SDP tokens on the DPT of the sentence are "FlyBack", "is", "Backup Software", "based", "on", and "Git". And their corresponding indices in the sentence are [1], [2], [7], [10], [11], and [12], respectively. The smallest index among these SDP tokens is [1], and the largest index is [12]. Hence the trimmed sentence is the subsequence from index [1] to index [12], i.e., "FlyBack is an open-source Backup Software for Linux based on Git", as shown in Figure 6.7.

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

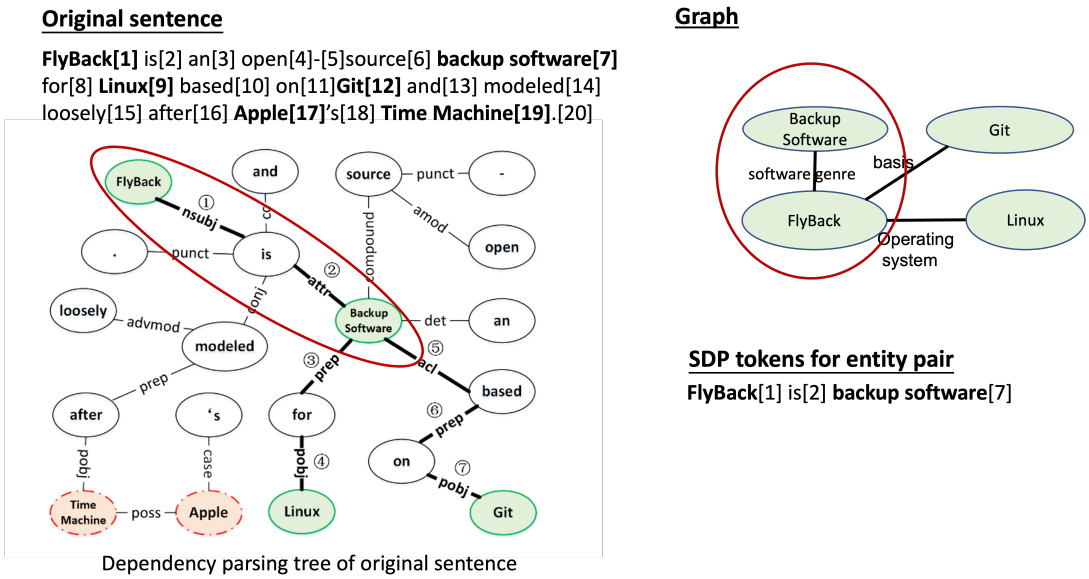


Figure 6.4: Tokens on the SDP between "FlyBack" and "Backup Software", and the corresponding subsequence in text

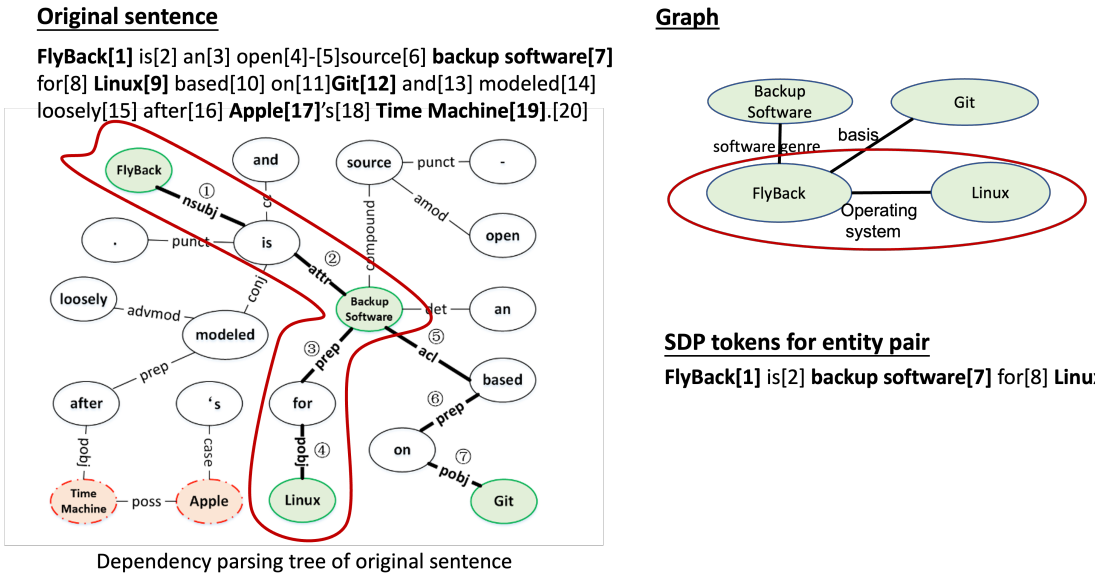


Figure 6.5: Tokens on the SDP between "FlyBack" and "Linux", and the corresponding subsequence in text

into a single token. Speficially, the two tokens `Backup` and `Software` were combined into token `BackupSoftware`.

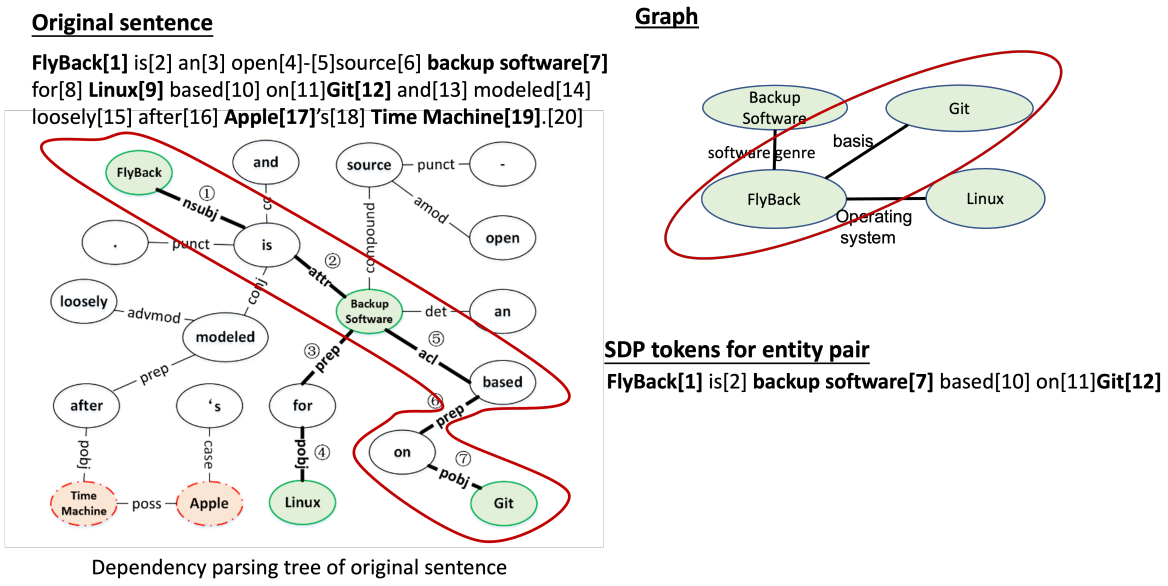


Figure 6.6: Tokens on the SDP between "FlyBack" and "Git", and the corresponding subsequence in text

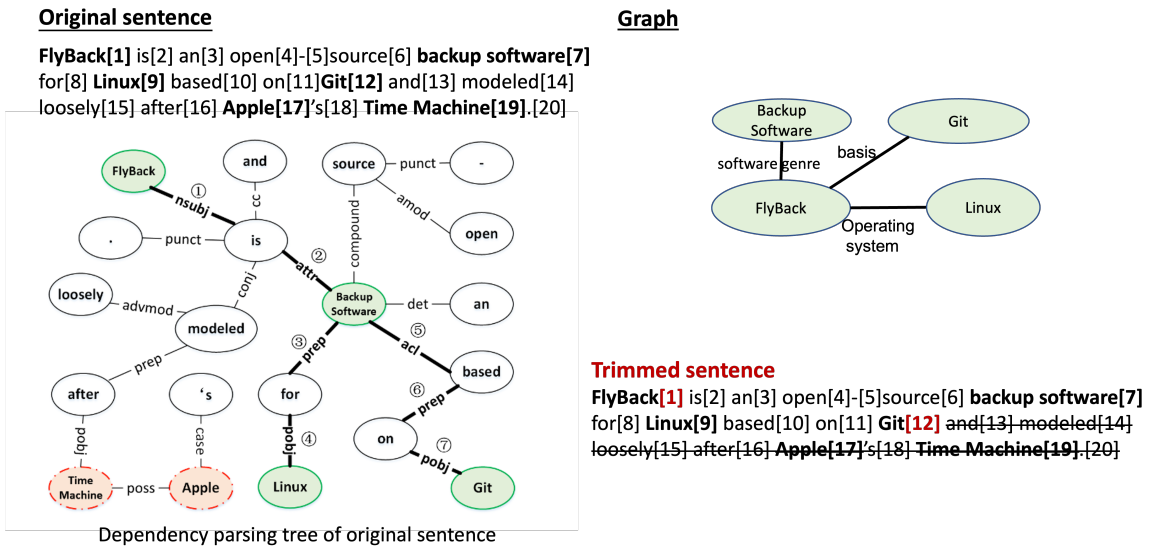


Figure 6.7: The graph and the final trimmed sentence

Figure 6.8 illustrates the whole workflow of building our graph-to-text model, which we call GraphNarrator. It covers the steps of dataset generation, sentence trimming for mitigating hallucination, and model fine-tuning. More specifically, the work-

flow comprises four groups of components, as shown in Figure 6.8: the entity aligner and the edge detector, which construct Freebase subgraphs from Wikipedia sentences; the sentence trimmer as a measure to mitigate hallucination; and the fine-tuning of a PLM. Within the entity aligner, the coreference resolution module replaces pronouns or aliases with the corresponding entities. For example, as Figure 6.8 shows, "the band" and "this release" are replaced with "Australian heavy metal band Lord" and "Return of the Tyrant", respectively. The Wikification module is applied to identify Wikipedia entities in sentences. These entities are then mapped to Freebase entities. Given the entities from a sentence, the edge detector further assembles a Freebase subgraph containing these entities. The triples of the subgraph are used to trim the coreference-resolved sentence. The resulting pairs of trimmed sentences and subgraphs are fed into the PLM for fine-tuning, creating a graph-to-text model. In the inference stage, the model takes a user-specified subgraph as input and outputs a generated sentence that describes it.

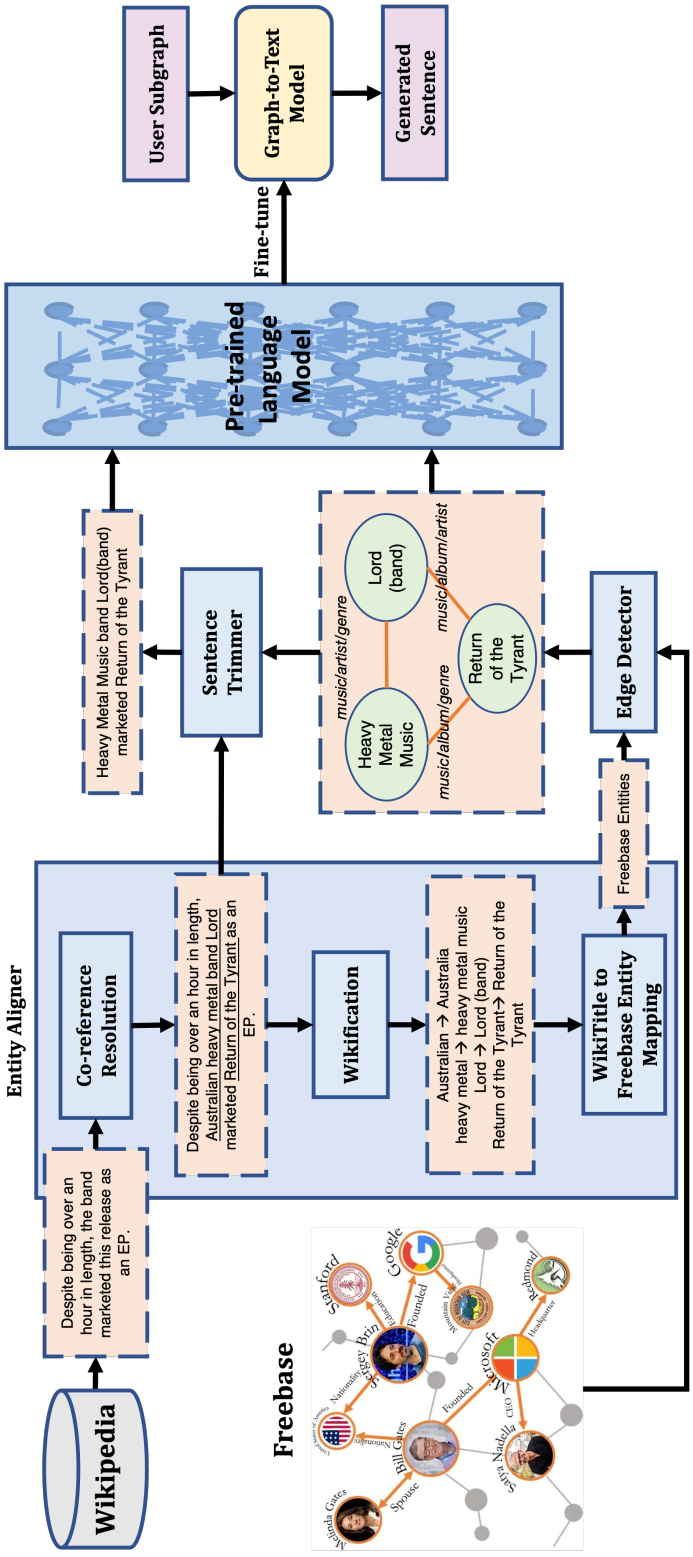


Figure 6.8: Workflow of GraphNarrator

CHAPTER 7

EXPERIMENTS & RESULTS

7.1 Datasets

We performed experiments on four datasets: GraphNarrative, TEKGEN (the large-scale, open-domain graph-to-text dataset that resembles ours the most), and WebNLG and DART (two human-annotated datasets). Detailed statistics about these and other datasets can be found in Table 4.2.

- GraphNarrative is partitioned into training, development and test sets in accordance with the process elaborated below. Each edge in Freebase belongs to a topic domain. Every instance in GraphNarrative, i.e., a (graph, sentence) pair, is assigned a domain, using the most frequent domain among the graph's edges. We then divided GraphNarrative into *seen* and *unseen* partitions according to the numbers of instance pairs in different domains. Domains with very few (less than 2,000) pairs were designated as unseen domains, while the remaining domains are seen. A full list of the seen and unseen domains is provided below. All instances in unseen domains go to the test set, thereby making them "unseen" from the training and development sets. In the seen partition, 90%, 5% and 5% of the instances are allocated for training, development and test, respectively. This resulted in 7,880,214 instances in the training set, 437,514 in the development set, and 451,906 in the test set, including 13,453 instances from unseen domains. Having unseen instances in the test set and having them only come from rare domains help us evaluate models' generalization ability. Unseen domains have limited presence in Wikipedia, as reflected in the aforementioned very few pairs from such domains in GraphNarrative. Given that Wikipedia is an important

corpus for PLMs in their pre-training process [Raffel et al., 2020, Lewis et al., 2020, Touvron et al., 2023], this helps ensure that the model has encountered only a small number of unseen instances during the pre-training stage of the PLMs. These unseen instances provide a more rigorous test of the model's ability to generalize to new and unfamiliar data.

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

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

• In TEKGEN, each instance pair contains a Wikipedia sentence and a Wikidata subgraph extracted from the sentence. We used the original training, development and test set partitions from [Agarwal et al., 2021]. To evaluate the effectiveness of the sentence trimming method on TEKGEN—another automatically collected dataset where fine-tuned PLMs also exhibit hallucination issues—we did not use all instances. This limitation arose due to the lack of mappings between entity names and their surface texts, as it is impossible to apply sentence trimming without such information. To maximize the utility of available instances, we used entity aliases sourced from TEKGEN, and we also leveraged regular expressions to identify time and peo-

ple's names. Consequently, we obtained 3,811,288 instances for training, 476,439 for development, and 484,958 for test, out of the original 6,310,061, 788,746, and 796,982 instances, respectively.

• In the standard WebNLG 2017 challenge dataset, each instance is composed of a graph from DBpedia and one or multiple sentences written by human annotations to describe the graph's content. The graph is generated from DBpedia using a content selection algorithm [Perez-Beltrachini et al., 2016]. Given an entity from a category, the algorithm selects the subgraph with the highest probability of being typical of that category and supporting the generation of a coherent text. Its test set is divided into the *seen* partition, which contains 10 DBpedia categories present in the training and development sets, and the *unseen* partition, which covers 5 categories absent from the training and development sets. We used the same partitioning as in the dataset.

• DART is a data-to-text dataset that consists of pairs of (triple-set, sentence) gathered from various sources, including WebNLG and E2E [Novikova et al., 2017]. It also includes (triple-set, sentence) pairs where the triple-sets are converted from the tables of WikiSQL [Zhong et al., 2017] and WikiTableQuestions [Pasupat and Liang, 2015], with sentences collected through crowdsourcing. We used the original partitioning of training, development and test sets in DART.

7.2 Human & Automatic Evaluation Metrics

7.2.1 Human evaluation metrics

Existing human evaluation metrics for graph-to-text generation primarily assess the linguistic quality of generated sentences, often neglecting to quantitatively address the issue of hallucination between the given graph and the generated text. Table 7.1

lists the human evaluation metrics used in both earlier and state-of-the-art studies. The commonly used metrics include *Semantics* (Does the text correctly represent the meaning of the data?), *Grammar* (Is the text free of spelling or grammatical errors?), *Fluency* (Does the text sound fluent and natural?), *Adequacy* (Does the text clearly express the data?), and *Correctness* (Is the output sentence semantically accurate?). These descriptions of the metrics are derived from the corresponding publications that utilize them, as in Table 7.1. Typically, each study employs two or three of these five metrics.

Paper	**Semantics**	**Grammar**	**Fluency**	**Adequacy**	**Correctness**
[Agarwal et al., 2021]	on a scale of 1-5	✗	on a scale of 1-5	✗	✗
[Gardent et al., 2017a]	on a scale of 1-3	on a scale of 1-3	on a scale of 1-3	✗	✗
[Colas et al., 2021]	✗	✗	✗	✗	✗
[Ribeiro et al., 2021]	on a scale of 1-7	✗	on a scale of 1-7	✗	✗
[Ribeiro et al., 2020]	✗	✗	on a scale of 1-5	on a scale of 1-5	✗
[Marcheggiani and Perez-Beltrachini, 2018]	✗	✗	✗	✗	✗
[Song et al., 2020]	✗	✗	✗	✗	✗
[Trisedya et al., 2018]	✗	on a scale of 1-3	on a scale of 1-3	✗	on a scale of 1-3
[Wang et al., 2021]	✗	✗	✗	✗	✗

Table 7.1: Human evaluation metrics used in graph-to-text studies

We evaluated the quality of both the GraphNarrative dataset and the sentences generated by models, focusing on 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 evaluated the *linguistic quality* of the sentences using a score averaged over the sentences, on a scale of 1-5 for each sentence: 5 (no errors), 4 (one error), 3 (two to three errors), 2 (four to five errors), and 1 (more than five errors). The "errors" in our evaluation refers to both grammatical errors and when a sentence is not fluent or uses awkward expressions. Thus, our linguistic quality score can be viewed as a combination of the metrics of grammar and fluency in Table 7.1.

7.2.2 Automatic evaluation metrics

For 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].

7.2.2.1 BLEU (Bilingual Evaluation Understudy)

The BLEU score, primarily used in machine translation, measures the similarity between a candidate (i.e., generated) sentence and one or more reference sentences. In our case, the reference sentences are the text of graph-text pairs in GraphNarrative, and the generated sentences are the model-inferred sentences given the graphs in GraphNarrative. The BLEU score is calculated as:

$$\text{BLEU} = BP \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$

where BP (brevity penalty) penalizes short generated sentences (defined below), p_n is the modified n-gram precision (also defined below), N is the highest order of n-grams considered, and w_n is the weight assigned to each modified n-gram precision and is often set as $w_n = \frac{1}{N}$.

In the BLEU score definition, the modified n-gram precision is defined as:

$$p_n = \frac{\sum_{C \in \text{Candidates}} \sum_{\text{g} \in NG(C)} \min(\text{Count(g)}, \text{Count}_{\text{ref}}(\text{g}))}{\sum_{C \in \text{Candidates}} \sum_{\text{g} \in NG(C)} \text{Count(g)}}$$

where n denotes the length of the n-grams being considered, such as unigrams (n=1), bigrams (n=2), trigrams (n=3), and so forth. Count(g) is the count of an n-gram g in the generated sentence, $\text{Count}_{\text{ref}}$(g) is the count of the n-gram in the reference sentences, and $NG(C)$ denotes the set of all n-grams in candidate sentence C. The modified n-gram precision measures the accuracy of n-grams (up to 4-grams [Popović, 2011]) in generated sentences by comparing them to those in reference sentences [Kondrak, 2005]. This metric accounts for the fact that the same n-gram may appear multiple times in a generated sentence but less in the reference sentences. To avoid over-penalizing repeated n-grams, the modified precision counts the minimum occurrence of an n-gram between each generated sentence and reference sentences.

The brevity penalty (BP) is defined as:

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-\frac{r}{c})} & \text{if } c \leq r \end{cases}$$

where c is the length of the generated sentence and r is the length of the reference sentence.

7.2.2.2 METEOR (Metric for Evaluation of Translation with Explicit ORdering)

The METEOR score is also commonly used in machine translation. Unlike BLEU, which relies solely on precision, METEOR incorporates both precision and recall, and it also applies stemming and synonymy matching. It is calculated by aggregating the scores across multiple pairs of generated-reference sentences, and the score for each pair is defined as:

$$\text{METEOR} = F_{\text{mean}} \cdot (1 - Pen)$$

where Pen is *fragmentation penalty* (defined below), and $F_{\text{mean}} = \frac{P \cdot R}{\alpha \cdot P + (1-\alpha) \cdot R}$ is the harmonic mean of precision ($P = \frac{m}{w_t}$) and recall ($R = \frac{m}{w_r}$), in which the parameter α

is typically set to 0.9 to weigh recall higher than precision, w_t is the number of words in the generated sentence, w_r is the number of words in the reference sentence, and m is the number of matches (which can be exact, stemmed, synonym, or paraphrase matches) between words in the generated and reference sentences.

The harmonic mean only accounts for congruity with respect to single words but not with respect to larger segments that appear in both reference and generated sentences. To address this, longer n-gram matches are used to compute a penalty p which gets higher values when more mappings between generated and reference sentences are not adjacent. To compute this penalty, unigrams are grouped into the fewest possible chunks, defined as sets of adjacent unigrams in both the generated sentence and the reference sentence. The longer the adjacent mappings between the candidate and the reference, the fewer chunks there will be. A generated sentence identical to the reference will result in just one chunk. The fragmentation penalty, Pen, is defined as:

$$Pen = \gamma \cdot \left(\frac{ch}{m}\right)^{\beta}$$

where ch is the number of chunks (contiguous matches) in the generated sentence, and the parameters γ and β are often set to 0.5 and 3.0, respectively [Banerjee and Lavie, 2005].

7.2.2.3 chrF++

The chrF++ score is particularly useful for languages with rich morphology and for translations where word boundaries are not well-defined. Similar to METEOR, it is aggregated over the scores for multiple pairs of generated-reference sentences, and the score for each pair is defined as:

$$F_\gamma = \frac{(1+\gamma^2) \cdot P \cdot R}{\gamma^2 \cdot P + R}$$

where γ is a parameter that balances the importance between precision (R) and recall (R). If $\gamma = 1$, they are equally important. Previous work [Popović, 2015] reported that the best γ parameter is 2 in terms of Kendall's τ segment-level correlation with human relative rankings.

The precision and recall in chrF++ are defined by combining character n-gram precision and recall (P_{chr} and R_{chr}) and word n-gram precision and recall (P_{word} and R_{word}), as follows:

$$P = (1-\beta) \cdot P_{chr} + \beta \cdot P_{word}$$

$$R = (1-\beta) \cdot R_{chr} + \beta \cdot R_{word}$$

where the parameter β is often set to 0.5. Given a generated sentence and a sentence, the character n-gram precision (typically up to 6-grams [Popović, 2015]) is the percentage of n-grams in the generated sentence that have a match in the reference sentence on the character level. The character n-gram recall is the percentage of n-grams in the reference sentence that are also present in the generated sentence on the character level. The word n-gram precision and recall are similar to their character level counterparts, except that the matches are assessed on the word level instead of character level.

7.3 Experiment and Evaluation Results

7.3.1 GraphNarrative dataset quality

Three human annotators evaluated the quality of the graph-sentence pairs in GraphNarrative. We randomly chose 100 pairs, where each sentence has the original version and the trimmed version using the algorithm in Chapter 6. The total 200

pairs were then shuffled so that annotators cannot tell whether a sentence is original or not. Each human annotator scored all 200 pairs using the metrics in Section 7.2, and their scores were averaged.

ST	Hallucinated Entities	Missed Entities	Hallucinated Relations	Missed Relations	Linguistic Quality
w/o	1.163	0.003	1.340	**0.040**	**4.793**
w/	**0.306**	0.003	**0.453**	0.083	4.613

Table 7.2: Human evaluation of GraphNarrative quality

Table 7.2 presents the results. In this and subsequent tables, sentence trimming is denoted ST. The average hallucinated entities and relations [1] per graph-sentence pair are 1.163 and 1.340, respectively. This reflects the challenges in graph-to-text alignment and the source of hallucination, as explained in Chapter 4. Applying sentence trimming reduced these numbers to 0.306 entities and 0.453 relations, clearly showing its effectiveness in enhancing graph-text alignment. On the other hand, when graphs were extracted from corresponding sentences to form GraphNarrative, information not present in the sentences was seldom introduced into the graphs, as reflected in the small missed entities and relations, both less than 0.1. Sentence trimming only slightly increased missed relations from 0.040 to 0.083, showing insignificant side effects of removing from sentences information covered in corresponding extracted graphs. With regard to linguistic quality, while sentence trimming led to a slight decline in the score, the difference (4.793 vs. 4.613) is not substantial.

[1] We used the metrics defined in Section 7.2 for evaluating both dataset quality and model output. This may cause confusions in understanding the measures in the context of dataset quality—a *hallucinated* entity refers to an entity from the original sentence that is *missed* in the corresponding extracted graph! We decided to tolerate this potential confusion for the sake of consistent metric definition.

Model	ST	BLEU			METEOR			chrF++		
		all	seen	unseen	all	seen	unseen	all	seen	unseen
TinyLLaMA	w/o	10.61	-	-	23.37	-	-	30.34	-	-
TinyLLaMA	w/	14.13	-	-	**34.86**	-	-	42.46	-	-
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 7.3: Model performance on GraphNarrative

7.3.2 Model performance on GraphNarrative

We fine-tuned TinyLLaMA (1.1B parameters) and various T5 (small: 60M parameters, base: 220M parameters, and large: 770M parameters) and BART (base: 140M parameters, large: 400M parameters) models on GraphNarrative 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 employed a linearly decreasing learning rate schedule without warm-up and set the maximum target text length to 384 tokens. Our implementation of BART and T5 models was based on the work by Ribeiro et al. [2021], which adapted PLMs from Hugging Face [Wolf et al., 2019] for graph-to-text. Since [Ribeiro et al., 2021] did not include the TinyLLaMA model, we adapted TinyLlama-1.1B-Chat-v1.0 from Hugging Face [Wolf et al., 2019] and implemented the same preprocessing steps as described in [Ribeiro et al., 2021]. The automatic evaluation results of different models on GraphNarrative are in Table 7.3. BART and T5 exhibited similar performance patterns. TinyLLaMA performed significantly lower in BLEU and chrF++ compared to other models. However, it achieved the highest METEOR scores. This suggests that TinyLLaMA may handle sentence structure and

semantic interpretation differently than BART and T5, warranting further investigation. Overall, fine-tuning the T5-large model yielded the best performance across most metrics, consistent with findings on WebNLG in [Ribeiro et al., 2021, Wang et al., 2021].

7.3.3 GraphNarrative in enhancing generalization ability

To assess if GraphNarrative may enhance PLMs' generalization ability, we conducted both zero-shot learning and fine-tuning experiments employing GN-T5 and GNST-T5 on WebNLG and DART, where GNST-T5 denotes the fine-tuned T5-large model on GraphNarrative with sentence trimming, and GN-T5 denotes the counterpart without sentence trimming. They are also compared with the original T5-large model as a point of reference.

Zero-shot results. For zero-shot learning, we directly applied the above-mentioned three models on the test sets of WebNLG and DART. The results are in Table 7.4. The results reveal that fine-tuning PLM on GraphNarrative substantially improves its generalization capabilities.

Model	**WebNLG**			**DART**		
	BLEU	**METEOR**	**chrF++**	**BLEU**	**METEOR**	**chrF++**
T5-large	4.01	9.54	24.64	3.44	7.93	23.17
Filter-T5	19.81	30.36	54.01	16.15	27.53	48.69
GN-T5	21.38	31.82	**56.83**	19.35	27.35	50.41
GNST-T5	**27.60**	**32.27**	56.81	**19.42**	**28.07**	**50.96**

Table 7.4: Zero-shot performance of models on WebNLG and DART test sets

Fine-tuning results. We subjected the three models to further fine-tuning on WebNLG and DART for 100 epochs with an early stopping patience of 20 epochs, while keeping other hyperparameters consistent with those in 7.3.2, Section 7.3. No trimming was performed on WebNLG and DART, as their sentences were authored

Model	BLEU			METEOR			chrF++		
	all	seen	unseen	all	seen	unseen	all	seen	unseen
[Gardent et al., 2017a]	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 7.5: Performance comparison of different graph-to-text models on WebNLG test set

by human annotators, with very few hallucinated or missed entities and relations. Table 7.5 compares the performance of different graph-to-text models on WebNLG test set, including the reprint of the results from seven prior studies. Gardent et al. [2017a] proposed the WebNLG 2017 challenge and provided a baseline model based on LSTM; Marcheggiani and Perez-Beltrachini [2018] developed a graph convolutional encoder for graph-to-text generation; Ferreira et al. [2019] compared various pipeline and end-to-end architectures and demonstrating that Transformer-based models perform effectively; Ribeiro et al. [2020] proposed a graph neural network (GNN)-based encoder to enhance the capture of local and global node contexts for text generation from knowledge graphs; Aghajanyan et al. [2021] integrated hyper-text into PLMs and reported improved performance; Ribeiro et al. [2021] was the first to investigate fine-tuning PLMs for graph-to-text generation; Wang et al. [2021] further enhanced the performance of PLMs by initially fine-tuning them on Wikipedia before fine-tuning on the WebNLG dataset. GNST-T5 fine-tuned on WebNLG outperformed others on most metrics, particularly in the unseen category. This improvement suggests that GraphNarrative enhances the generalization ability of PLMs. Table 7.6 shows the fine-tuning results on DART test set. The model performance improvement by sentence trimming is not obvious. This is further discussed in Section 7.3.4.

Model	BLEU	METEOR	chrF++
T5-large	50.38	39.98	68.06
GN-T5	**50.53**	39.99	68.15
GNST-T5	50.51	**40.07**	**68.23**

Table 7.6: Fine-tuning results on DART 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 7.7: Performance of fine-tuning BART-large and T5-large on the TEKGEN dataset

Dataset	ST	BLEU			METEOR			chrF++		
		all	seen	unseen	all	seen	unseen	all	seen	unseen
TEKGEN	w/o	60.43	65.49	54.32	44.06	46.04	41.90	75.73	78.83	70.13
TEKGEN	w/	60.82	65.42	55.12	44.25	46.13	42.18	76.16	79.11	72.35
GraphNarrative	w/o (GN-T5)	60.26	65.44	54.06	44.08	45.90	41.98	75.83	79.02	72.35
GraphNarrative	w/ (GNST-T5)	61.46	66.49	55.35	44.30	46.23	42.08	76.20	79.35	72.76

Table 7.8: Models' performance on WebNLG test set, when fine-tuned with TEKGEN or GN and further fine-tuned with WebNLG

7.3.4 Ablation study of sentence trimming

We demonstrate the effectiveness of sentence trimming in improving model performance on GraphNarrative, TEKGEN, WebNLG, and DART by fine-tuning PLMs with and without sentence trimming, respectively. (1) For GraphNarrative, we fine-tuned T5, BART, and TinyLLaMA models using the setup described in 7.3.2, Section 7.3. (2) For TEKGEN, we fine-tuned the T5-large and BART-large models using the serialized triples from [Agarwal et al., 2021], with the same hyperparameters as in 7.3.2, Section 7.3. (3) For WebNLG and DART, we conducted zero-shot learning and fine-tuning experiments as described in 7.3.3, Section 7.3. (4) Additionally, on the WebNLG dataset, we carried out further fine-tuning of the T5-large model fine-tuned on TEKGEN in (2), applying the same hyperparameters as in 7.3.3, Section 7.3.

The results of (1) and (2) are in Tables 7.3 and 7.7. The metrics (BLEU, METEOR, chrF++) consistently improve with sentence trimming, further verifying the efficacy of sentence trimming. The results of (3) are in Tables 7.4, 7.8 and 7.6, and Table 7.8 also shows the results of (4). In these results, the fine-tuned PLMs on GraphNarrative and TEKGEN with sentence trimming consistently outperformed their non-trimming counterparts. These findings underscore the effectiveness of sentence trimming in enhancing PLM performance. It is worth noting that, as Tables 7.8 and 7.6 show, on human-annotated WebNLG and DART the models did not gain much from sentence trimming after they are fine-tuned on these datasets. The main reason is that human-annotated datasets generally have well-aligned graph-text pairs and thus cannot be substantially improved by trimming.

7.3.5 Sentence trimming in mitigating hallucination

We randomly sampled 100 graphs from GraphNarrative test set, along with the corresponding sentences generated by GNST-T5 and GN-T5. We shuffled the 200 pairs and used three human evaluators to score the pairs, in the same fashion as in 7.3.1, Section 7.3. The results are in Table 7.9, which shows a reduction of 1.4 hallucinated entities and 1.0 hallucinated relations per instance from GN-T5 to GNST-T5, suggesting that sentence trimming effectively mitigates hallucinations. Furthermore, sentences generated by both models exhibit on average less than 0.07 missed entities and 0.38 missed relations per instance.

Table 7.10 illustrates the sentences generated by GNST-T5 and GN-T5 for a few input graphs. GN-T5 tends to fabricate facts that are incorrect or non-existent in the real world (e.g., `Arthur Morry`'s age of death, the renaming of the `US Naval Academy`, and `Goldie Gets Along`'s year of release) or not present in input graphs (e.g., `Goldie Gets Along`'s genre). In contrast, in the illustrated examples, GNST-T5

generated fluent sentences without fabricating facts, barring a phrase instead of a complete sentence for the second example.

ST	Hallucinated Entities	Missed Entities	Hallucinated Relations	Missed Relations	Linguistic Quality
w/o	1.643	0.063	1.363	**0.240**	**4.613**
w/	**0.260**	**0.056**	**0.300**	0.370	4.356

Table 7.9: Human evaluation of sentences generated by T5-large with and without sentence trimming

Regarding linguistic quality, sentences generated by GNST-T5 received slightly lower scores than GN-T5. Nevertheless, these scores remain acceptable, with on average less than one error per instance.

To further understand the errors introduced by sentence trimming, we analyzed the human evaluation results of the generated sentences by GN-T5 and GNST-T5 and focused on the cases where the linguistic quality score became lower with sentence trimming. We found that 51.9% of such cases involve location-related relationships such as *location contains*, *born place*, *death place*, and so on.

We analyzed how we may mitigate such errors, particularly grammar errors, and we found it will be challenging to modify the trimming algorithm to reduce grammar errors since oftentimes the sentence itself does not have a verb between the entities we extracted to the graph. For instance, consider the following sentence from Wikipedia: "The Dallas Mavericks (often referred to as the Mavs) are an American professional basketball team based in Dallas, Texas." The corresponding graph aligned with Freebase is (`Texas`, *location contains*, `Dallas`). [2] The tokens between `Dallas` and

[2] "Dallas Mavericks" is not mapped to Freebase since the corresponding entity is not present in Freebase. In more general cases, except for the incompleteness of a knowledge graph, this missing mapping could also be caused by errors in the graph-text alignment algorithm.

Input Graph	Sentence Generated by GN-T5	Sentence Generated by GNST-T5
(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).
(Huangpu River, cities, Shanghai)	The Huangpu River Bridge () is a bridge over the Huangpu River in Shanghai, China.	Huangpu River in Shanghai
(Los Angeles Rams, team player, Jerome Bettis) (Pittsburgh Steelers, team player, Jerome Bettis)	Jerome Bettis was drafted by the Los Angeles Rams in the third round of the 2009 NFL Draft, and signed by the Pittsburgh Steelers in the fourth round of the 2009 NFL Draft.	Jerome Bettis signed with the Los Angeles Rams and Pittsburgh Steelers
(Hertha BSC, team player, Marc Arnold) (Hertha BSC, team league, Bundesliga)	Hertha BSC in a 3–0 win against SV Werder Bremen.	Marc Arnold signed a one-year contract with Bundesliga club Hertha BSC.
(Trances/Drones, genre, Ambient music) (Trances/Drones, artist, Robert Rich(musician)) (Robert Rich (musician), genre, Ambient music)	Trances/Drones is the second studio album by Ambient music producer Robert Rich (musician).	Trances/Drones is an Ambient music album by Robert Rich (musician).
(Canadian Pacific Railway, headquarters country, Canada) (British Columbia, contains, Penticton) (Canada, contains, British Columbia) (Canada, country citytown, Penticton)	The Canadian Pacific Railway (CPR) is a Canadian railway company based in Penticton, British Columbia, Canada.	Canadian Pacific Railway (CPR) is a Canadian railway company headquartered in Penticton, British Columbia, Canada.
(Georgina Cassar, nationality, United Kingdom) (2010 Commonwealth Games, country, India) (India, contains, Delhi) (2012 Summer Olympics, olympics country, India) (2012 Summer Olympics, olympics athlete, Georgina Cassa) (2012 Summer Olympics, locations, London) (2012 Summer Olympics, olympics country, United Kingdom) (United Kingdom, contains, London) (Gibraltar Cassar, country, United Kingdom)	Georgina Cassar competed at the 2010 Commonwealth Games in Delhi, India and the 2012 Summer Olympics in London, United Kingdom.	Georgina Casar competed at the 2010 Commonwealth Games in Delhi, India and the 2012 Summer Olympics in London, United Kingdom.

Table 7.10: Comparison of generated sentences with and without sentence trimming for sample input graphs

`Texas` on the shortest dependency path (SDP) of the sentence are "Dallas", "," and "Texas", resulting in the verb-absent trimmed sentence "Dallas, Texas" for the graph (`Texas`, *location contains*, `Dallas`). This issue is not unique; we found that 50% (23 out of 46) of location-related trimmed sentences with grammar errors are caused by the absence of verbs.

One plausible direction in addressing this grammar issue is to utilize human-annotated sentences containing location-related relations from the WebNLG dataset. We further fine-tuned the GNST-T5 model, using these instances. The enhanced model demonstrates improved performance, producing grammatically correct sentences when tested on the same graphs. Figure 7.1 shows this process. For the same graph (`Annapolis Maryland`, *location contains*, `US Naval Academy`), the further fine-tuned GNST-T5 on WebNLG dataset produced grammarly correct sentence "US Naval Academy is located in Annapolis Maryland." Although this idea shows promises in this example, we need more systematic human evaluation in the future.

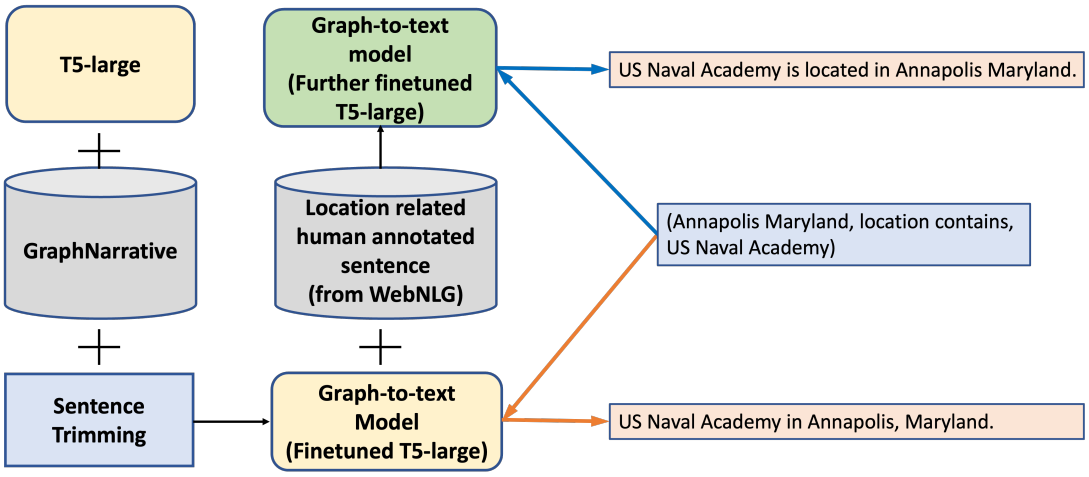


Figure 7.1: Further fine-tune on human-annotated dataset to reduce grammar errors

7.3.6 Limitations of star graph datasets

As explained in Chapter 1, existing large-scale datasets such as TEKGEN contain predominantly star graphs. We used GraphNarrative to investigate the limitations of star graph datasets.

More specifically, we separated the graph-sentence pairs in GraphNarrative into *star* instances (with star graphs) and *non-star* instances (without star graphs). We excluded instances with two or three entities, as they could be considered as both paths and stars. Table 7.11 provides the distributions of these two types of instances. The number of non-star instances in all three sets is approximately 3.5 times as many as the star instances. To help ensure a fair comparison, we randomly selected an equal number of non-star instances as the star instances for each of the three sets, e.g., there are 290,047 star graphs and the same number of non-star graphs in the training set of our prepared dataset.

Using the dataset prepared this way, we fine-tuned T5-large model with and without sentence trimming for 10 epochs under early stopping patience 5, using the same other hyperparameters as in 7.3.2, Section 7.3. The results are in Table 7.12. Across the board, models trained using star instances exhibited the highest performance when tested using star instances too, and similarly regarding non-star instances. Furthermore, models trained on non-star instances and tested on star instances tended to outperform models trained on star instances and tested on non-star instances. These results indicate that which are commonly encountered in real-world applications. Fine-tuning PLMs on diverse graph shapes enhances their generalization capability.

Set	Star	Non-star	All
train	290,047	1,022,303	1,312,350
dev	16,192	56,612	72,804
test	16,425	58,517	74,942
all	322,664	1,137,432	1,460,096

Table 7.11: Number of star and non-star instances in GraphNarrative

ST	Train	Test	BLUE	METEOR	chrF++
w/o	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
w/	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 7.12: Model performance, star vs. non-star graphs

7.3.7 Comparing sentence trimming with filtering

We compared sentence trimming with a similar but different filtering method proposed in [Ma et al., 2022]. Their method also aimed to reduce disparities in datasets as a way of mitigating hallucination. However, different from our approach which aligns sentences better with input graphs by trimming away portions of sentences, the filtering method removes graph-text pairs from the DART dataset where the ROUGE-1 similarity score between the graph and the text is below 0.8.

We applied the same filtering method on GraphNarrative. Table 7.13 provides a breakdown of the remaining instances after filtering using different thresholds. A rel-

atively low threshold of 0.3 removed 43.32% of the instances in GraphNarrative. When we raised the threshold to 0.8, almost all instances were eliminated. In comparison, the threshold of 0.8 applied on DART allowed for retaining 88% of its instances. This is because the human-annotated DART has well-aligned graph-text pairs.

Threshold	**ST**	**No filter**	**0.8**	**0.5**	**0.3**
Dev	w/o	437514	74 (0.0169%)	14952 (3.416%)	142974 (32.678%)
Dev	w/	437514	791 (0.1806%)	71565 (15.357%)	256357 (58.593%)
Test	w/o	451906	65 (0.0144%)	15843 (3.507%)	148666 (32.892%)
Test	w/	451906	931 (0.206%)	75445 (16.689%)	266094 (58.886%)
Train	w/o	7880214	1328 (0.0168%)	269541 (3.418%)	2579921 (32.740%)
Train	w/	7880214	14787 (0.1877%)	1293864 (16.405%)	4621448 (58.659%)
Total	w/o	8769634	1467 (0.017%)	300336 (3.43%)	4970561 (56.68%)
Total	w/	8769634	16509 (0.1883%)	1382074 (15.76%)	7445899 (84.88%)

Table 7.13: Number of remaining instances after filtering using different thresholds of ROUGE-1 similarity scores

We compared sentence trimming with filtering using the 269,541 instances left in the training set and 71,565 in the development set, under threshold 0.5. We fine-tuned the T5-large model for 10 epochs with early stopping patience 5, using the same other hyperparameters as on the full dataset. The number of training steps is different from the full dataset because this subset is about 30 times smaller. We used early stopping to avoid overfitting. Then we used the resulting model, which we call Filter-T5, for zero-shot prediction on WebNLG and DART test sets. The results are shown in Table 7.4. GNST-T5 slightly outperformed Filter-T5. To understand this, we compared the statistics of the filtered dataset and the full dataset (and thus the dataset after sentence trimming since trimming does not alter the graphs in the dataset), as in Table 7.14. The filtered dataset exhibits a significant reduction in size and diversity in terms of number of distinct entities, relations, triples and shapes. We

conjecture that this contributes to its performance degeneration in comparison with GNST-T5.

Dataset	Entities	Triples	Relations	Star Graphs	Shapes
Filtered	307,590	437,519	974	47%	597
GraphNarrative	1,853,752	15,472,249	1,724	22%	7,920

Table 7.14: Statistics of GraphNarrative and its filtered dataset

7.3.8 Performance of GNST-T5 and GN-T5 by input size

#Triples	BLEU (GN-T5)	BLEU (GNST-T5)
1	13.60	25.97
2	19.90	27.53
3	24.07	29.73
4	30.51	32.62
5	32.95	35.22
6	38.81	39.73
7	42.74	41.55
8	42.23	42.10
9	55.73	51.83
10	45.84	48.08
11	41.72	42.71
12	38.09	39.33
13	41.77	43.27
14	33.47	36.95
15	34.04	38.44

Table 7.15: Distribution of GNST-T5 and GN-T5 model performance in BLEU scores on GraphNarrative test set

Table 7.15 shows the performance of GNST-T5 and GN-T5 in BLEU scores on graphs of varying sizes, i.e., number of triples. The results help gauge whether the models generalize well for long inputs. Notably, the performance of both models on extended inputs is better than or on par with their performance on shorter inputs.

7.3.9 Comparing two linearization methods

Table 7.16 compares the performance of the fine-tuned T5-large model using two different linearization methods (base linearization and structured linearization, introduced in Section 5.3) with the same experiment setting in Section 7.3.2. The differences between basic and structured linearizations are minor in terms of performance metrics. In understanding the observed small influence of the linearization method on model performance in graph-to-text, we hypothesize a couple of possible explanations, as follows.

ST	**Linearization**	**BLEU**			**METEOR**			**chrF++**		
		all	seen	unseen	all	seen	unseen	all	seen	unseen
w/o	Basic	22.22	22.26	20.41	17.16	17.23	15.02	36.78	36.95	31.40
w/o	Structured	22.40	22.40	22.08	17.21	17.28	15.09	36.86	37.02	31.44
w/	Basic	45.12	45.16	43.40	24.77	24.84	22.54	50.44	50.60	45.21
w/	Structured	44.98	45.05	42.00	24.74	24.81	22.44	50.41	50.57	45.15

Table 7.16: Model performance on GraphNarrative test set when fine-tune T5-large with two different linearization methods

Robustness of the model architecture: Advanced models such as T5-large are highly capable of handling complex and varied input structures. Their architecture and training methods are designed to extract and utilize essential information effectively, regardless of slight variations in how input data is presented. This may mean that differences in linearization methods do not necessarily have significant impact on the model's ability to understand and generate text from graph representations.

Limitations of metric sensitivity: The metrics used (BLEU, METEOR, chrF++) may not be sufficiently sensitive to capture the subtle nuances introduced by different linearization methods. These metrics primarily measure token and character n-gram overlap between generated sentences and reference sentences. Given this limitation, although the generated sentences produced by models with different linearization

methods may differ, it is possible the differences are not effectively captured by these metrics.

CHAPTER 8

CONCLUSION

The contributions of the work are as follows.

• A new dataset, GraphNarrative, that fills the gap between existing datasets and large-scale real-world settings. This dataset is specifically designed to address the limitations of current datasets by incorporating more diverse and complex data, thereby providing a more realistic and challenging benchmark for graph-to-text generation models. By doing so, it facilitates the development and evaluation of models in environments that closely resemble real-world applications.

• We are the first to quantify hallucinations produced by graph-to-text models. Hallucinations refer to the generation of content that is not present in the input data. Our work pioneers to measure these hallucinations within text generated from graphs. By developing a set of quantitative metrics, we can objectively assess the extent and frequency of hallucinations, providing a clear framework for comparison across different models and approaches.

• A novel approach, sentence trimming, to hallucination mitigation. This approach is both innovative and practical, offering a new method for research aimed at enhancing the fidelity and reliability of graph-to-text generation systems. By eliminating parts of sentences that contribute to hallucinations, sentence trimming effectively reduces the occurrence of unsupported content while maintaining the overall coherence and informativeness of the generated text.

• Comprehensive experiments and evaluations that verify the quality and utility of GraphNarrative, as well as the effectiveness of sentence trimming. Our extensive

experimental setup includes a variety of evaluation metrics and scenarios, ensuring that our findings are robust and widely applicable. The results demonstrate not only the high quality of the GraphNarrative dataset but also the improvements in model performance achieved through sentence trimming. These experiments include comparisons with existing datasets and models, highlighting the advantages of our proposed dataset and approach in various contexts and applications.

CHAPTER 9

LIMITATIONS

• The creation of GraphNarrative and the sentence trimming method leverage an existing mapping between the knowledge graph entities and Wikipedia entities. Given other text corpora and knowledge graphs, creating such a mapping is a non-trivial undertaking that often requires named entity recognition and disambiguation techniques.

• The sentence trimming approach may introduce grammatical errors into generated sentences.

• The method focuses on describing the content of an input graph only, without considering context information such as neighboring entities in the knowledge graph. Such extra information may be preferred by a user given certain application contexts or may make the input graph's narration more natural.

• The creation of GraphNarrative does not consider multiary relationships in knowledge graphs. More specifically, the Freebase used in our work is a version in which multiary relationships were converted into binary relationships [Shirvani-Mahdavi et al., 2023]. In general, there is a lack of inquiry into multiary relationships in graph-to-text models. To the best of our knowledge, the only work in this area that discusses such multiary relationships is [Agarwal et al., 2021] and they also converted multiary relationships into binary ones.

• A couple of studies [Agarwal et al., 2021, Wang et al., 2021] attempted to address hallucination by further fine-tuning PLMs on WebNLG after fine-tuning on noisier automatically-extracted datasets. It will be informative to conduct a human

evaluation comparison between their approaches and the sentence trimming method proposed in our work. Similarly, our future work includes a human evaluation comparison with the filtering-based method [Ma et al., 2022] which we empirically compared with in Section 7.3.7.

- The sentence trimming algorithm only removes irrelevant portions from the beginning and the end of a sentence, leaving the token sequence in the middle intact. It is possible the middle portion also contains tokens irrelevant to the input graph.

CHAPTER 10

Ethics Statement

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

Given that our research focuses on producing natural language descriptions of knowledge graphs, we are particularly aware of the potential misuse of our method for the generation of false, deceptive, biased or unfair contents. Particularly, our sentence trimming method aims to minimize such potential misuse by aiding in reducing hallucinations.

We also recognize that natural language descriptions generated using our dataset and algorithm can be repurposed in various ways. We firmly urge users and developers to use this content responsibly, particularly with respect to intellectual property rights. Furthermore, we recommend users clearly label AI-generated content, promoting transparency and trust.

Our GraphNarrative dataset uses publicly available data, particularly Freebase and Wikipedia, which do not contain information that violates anyone's privacy to the best of our knowledge.

Our reliance on Wikipedia may inadvertently introduce bias, as Wikipedia content can reflect the views of its contributors. We are also aware this potential bias could be more intense in less commonly spoken languages, where the number of contributors might be limited if one applies our approach to such languages.

Bibliography

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Oshin Agarwal, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. 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, 2021.

Armen Aghajanyan, Dmytro Okhonko, Mike Lewis, Mandar Joshi, Hu Xu, Gargi Ghosh, and Luke Zettlemoyer. HTLM: Hyper-text pre-training and prompting of language models. In *International Conference on Learning Representations*, volume abs/2107.06955, 2021.

Abdulla Alsharhan. Natural language generation and creative writing a systematic review. *Journal of International Journal of Advances in Applied Computational Intelligence*, 1(1):69–90, 2022.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 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, 2007.

Satanjeev Banerjee and Alon Lavie. 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, 2005.

Bradley R Bebee, Daniel Choi, Ankit Gupta, Andi Gutmans, Ankesh Khandelwal, Yigit Kiran, Sainath Mallidi, Bruce McGaughy, Mike Personick, Karthik Rajan, et al. Amazon neptune: Graph data management in the cloud. In *International Semantic Web Conference (P&D/Industry/BlueSky)*, 2018.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. A neural probabilistic language model. *Advances in Neural Information Processing Systems*, 13, 2000.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 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, 2008.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*, 2015.

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

Thiago Castro Ferreira, Chris van der Lee, Emiel van Miltenburg, and Emiel Krahmer. 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, 2019.

Alison J Cawsey, Bonnie L Webber, and Ray B Jones. Natural language generation in health care. *Journal of the American Medical Informatics Association*, 4(6): 473–482, 1997.

Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C Hsieh, Deborah A Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E Gruber. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems*, 26(2):1–26, 2008.

Valeriy Chernenkiy, Yuriy Gapanyuk, Anatoly Nardid, Maria Skvortsova, Anton Gushcha, Yuriy Fedorenko, and Richard Picking. Using the metagraph approach for addressing RDF knowledge representation limitations. In *2017 Internet Technologies and Applications*, pages 47–52. IEEE, 2017.

Ching-Yao Chuang, Joshua Robinson, Yen-Chen Lin, Antonio Torralba, and Stefanie Jegelka. Debiased contrastive learning. *Advances in Neural Information Processing Systems*, 33:8765–8775, 2020.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*, 2020.

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

Anthony Colas, Ali Sadeghian, Yue Wang, and Daisy Zhe Wang. 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*, 2021.

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

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

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

Orri Erling and Ivan Mikhailov. RDF Support in the Virtuoso DBMS. In *Networked Knowledge-Networked Media: Integrating Knowledge Management, New Media Technologies and Semantic Systems*, pages 7–24. Springer, 2009.

Michael Färber. *Semantic Search for Novel Information*. IOS Press, 2017.

Thiago Castro Ferreira, Chris van der Lee, Emiel Van Miltenburg, and Emiel Krahmer. 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, 2019.

Jack FitzGerald, Shankar Ananthakrishnan, Konstantine Arkoudas, Davide Bernardi, Abhishek Bhagia, Claudio Delli Bovi, Jin Cao, Rakesh Chada, Amit Chauhan, Luoxin Chen, et al. Alexa teacher model: Pretraining and distilling multi-billion-parameter encoders for natural language understanding systems. In *Proceedings*

of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pages 2893–2902, 2022.

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

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

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software*, pages 1–6. Association for Computational Linguistics, 2018.

Hitesh Golchha, Mauajama Firdaus, Asif Ekbal, and Pushpak Bhattacharyya. Courteously yours: Inducing courteous behavior in customer care responses using reinforced pointer generator network. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 851–860. Association for Computational Linguistics, 2019.

Rufus H Gouws. The use of an improved access structure in dictionaries. *Lexikos*, 11, 2001.

José Rolando Guay Paz and José Rolando Guay Paz. Introduction to Azure COSMOS DB. *Microsoft Azure Cosmos DB Revealed: A Multi-Model Database Designed for the Cloud*, pages 1–23, 2018.

Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy)*, 2008.

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

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

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

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 328–339, 2018.

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

Andreas Kaplan and Michael Haenlein. Siri, Siri, in my hand: Who's the fairest in the land? On the interpretations, illustrations, and implications of artificial intelligence. *Business Horizons*, 62(1):15–25, 2019.

Diederik P Kingma and Jimmy Ba. Adam: a method for stochastic optimization. In *International Conference on Learning Representations*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *International Conference on Learning Representations*, 2014.

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

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 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, 2019.

Grzegorz Kondrak. N-gram similarity and distance. In *International Symposium on String Processing and Information Retrieval*, pages 115–126. Springer, 2005.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, 2017.

Wang-Chien Lee and Dik Lun Lee. Combining indexing technique with path dictionary for nested object queries. In *Proceedings of the 4th International Conference on Database Systems for Advanced Applications*, pages 107–114, 1995.

Leo Leppänen, Myriam Munezero, Mark Granroth-Wilding, and Hannu Toivonen. Data-driven news generation for automated journalism. In *Proceedings of the 10th*

International Conference on Natural Language Generation, pages 188–197. Association for Computational Linguistics, 2017.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 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, 2020.

Fei Li and Hosagrahar V Jagadish. Constructing an interactive natural language interface for relational databases. *Proceedings of the Very Large Data Base Endowment*, 8(1):73–84, 2014.

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

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

Peng Lin, Qi Song, Jialiang Shen, and Yinghui Wu. Discovering graph patterns for fact checking in knowledge graphs. In *International Conference on Database Systems for Advanced Applications*, pages 783–801, 2018.

Yang Liu and Mirella Lapata. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3730–3740, 2019.

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

Kaixin Ma, Hao Cheng, Xiaodong Liu, Eric Nyberg, and Jianfeng Gao. Open domain question answering with a unified knowledge interface. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 1605–1620, 2022.

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

Joseph F McCarthy and Wendy G Lehnert. Using decision trees for conference resolution. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1050–1055, 1995.

Justin J Miller. Graph database applications and concepts with Neo4j. In *Proceedings of the Southern Association for Information Systems Conference*, volume 2324, pages 141–147, 2013.

Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Surveys*, 56(2):1–40, 2023.

Linyong Nan, Dragomir Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, et al.

DART: Open-domain structured data record to text generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 432–447, 2021.

Rungsiman Nararatwong, Natthawut Kertkeidkachorn, and Ryutaro Ichise. Knowledge graph visualization: Challenges, framework, and implementation. In *2020 IEEE Third International Conference on Artificial Intelligence and Knowledge Engineering*, pages 174–178. IEEE, 2020.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206, 2017.

OpenAI. ChatGPT [large language model]. `https://openai.com/chatgpt`, 2022.

Jeff Z Pan. Resource description framework. In *Handbook on ontologies*, pages 71–90. Springer, 2009.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 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, 2002.

Raghavendra Pappagari, Piotr Zelasko, Jesús Villalba, Yishay Carmiel, and Najim Dehak. Hierarchical transformers for long document classification. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop*, pages 838–844. IEEE, 2019.

Panupong Pasupat and Percy Liang. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural*

Language Processing, pages 1470–1480. Association for Computational Linguistics, 2015.

Thomas Pellissier Tanon, Denny Vrandečić, Sebastian Schaffert, Thomas Steiner, and Lydia Pintscher. From Freebase to Wikidata: The great migration. In *Proceedings of the 25th International Conference on World Wide Web*, pages 1419–1428. Association for Computing Machinery, 2016.

Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. Semantics and complexity of SPARQL. *ACM Transactions on Database Systems*, 34(3):1–45, 2009.

Laura Perez-Beltrachini, Rania Sayed, and Claire Gardent. Building RDF content for data-to-text generation. In *Proceedings of the 26th International Conference on Computational Linguistics*, pages 1493–1502, 2016.

Maja Popović. Morphemes and POS tags for n-gram based evaluation metrics. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 104–107. Association for Computational Linguistics, 2011.

Maja Popović. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the 10th Workshop on Statistical Machine Translation*, pages 392–395. Association for Computational Linguistics, 2015.

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

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer

learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

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

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

Andrea Rossi, Denilson Barbosa, Donatella Firmani, Antonio Matinata, and Paolo Merialdo. Knowledge graph embedding for link prediction: A comparative analysis. *ACM Transactions on Knowledge Discovery from Data*, 15(2):1–49, 2021.

Julian Salazar, Davis Liang, Toan Q Nguyen, and Katrin Kirchhoff. Masked language model scoring. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2699–2712, 2020.

Martin Schmitt, Leonardo FR Ribeiro, Philipp Dufter, Iryna Gurevych, and Hinrich Schütze. Modeling graph structure via relative position for text generation from knowledge graphs. In *Proceedings of the 15th Workshop on Graph-Based Methods for Natural Language Processing*, pages 10–21, 2021.

Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31, 2018.

Nasim Shirvani-Mahdavi, Farahnaz Akrami, Mohammed Samiul Saeef, Xiao Shi, and Chengkai Li. Comprehensive analysis of Freebase and dataset creation for robust evaluation of knowledge graph link prediction models. In *International Semantic Web Conference*, pages 113–133. Springer, 2023.

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

Mirco Speretta and Susan Gauch. Personalized search based on user search histories. In *The 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 622–628. IEEE, 2005.

Dan Su, Yan Xu, Genta Indra Winata, Peng Xu, Hyeondey Kim, Zihan Liu, and Pascale Fung. Generalizing question answering system with pre-trained language model fine-tuning. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 203–211, 2019.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. LSTM neural networks for language modeling. In *Interspeech*, pages 194–197, 2012.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Bayu Distiawan Trisedya, Jianzhong Qi, Rui Zhang, and Wei Wang. 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, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

Mehul Nalin Vora. Hadoop-HBase for large-scale data. In *Proceedings of 2011 International Conference on Computer Science and Network Technology*, pages 601–605. IEEE, 2011.

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

Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F Wong, and Lidia S Chao. Learning deep transformer models for machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1810–1822, 2019.

Qingyun Wang, Semih Yavuz, Xi Victoria Lin, Heng Ji, and Nazneen Rajani. Stage-wise fine-tuning for graph-to-text generation. In *Proceedings of the The Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing Student Research Workshop*, pages 16–22, 2021.

Jianfeng Wen, Jianxin Li, Yongyi Mao, Shini Chen, and Richong Zhang. On the representation and embedding of knowledge bases beyond binary relations. In

Proceedings of the International Joint Conference on Artificial Intelligence, page 1300–1307, 2016.

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

World Wide Web Consortium. Resource Description Framework (RDF): Concepts and Abstract Syntax. `https://www.w3.org/TR/rdf11-concepts/`. Accessed: 2024-07-14.

Yikun Xian, Zuohui Fu, Shan Muthukrishnan, Gerard De Melo, and Yongfeng Zhang. 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, 2019.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. XLNet: Generalized autoregressive pretraining for language understanding. *Advances in Neural Information Processing Systems*, 32, 2019.

Shuzhou Yuan and Michael Färber. Evaluating generative models for graph-to-text generation. *arXiv preprint arXiv:2307.14712*, 2023.

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

Huibing Zhang, Junchao Dong, Liang Min, and Peng Bi. A BERT fine-tuning model for targeted sentiment analysis of chinese online course reviews. *International Journal on Artificial Intelligence Tools*, 29, 2020.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. TinyLLaMA: An open-source small language model. *arXiv preprint arXiv:2401.02385*, 2024.

Zhiping Zheng. Answerbus question answering system. In *Proceedings of the second international conference on Human Language Technology Research*, pages 399–404, 2002.

Victor Zhong, Caiming Xiong, and Richard Socher. Seq2SQL: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*, 2017.

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

BIOGRAPHICAL STATEMENT

Xiao Shi was born in Gaoyi, Shijiazhuang, Hebei, China in 1994. She received her B.E. degree in Electronic Information Engineering from Xidian University, China, in 2017, and her Ph.D. degree in Computer Science from The University of Texas at Arlington in Computer Science in 2024. Her research interests include natural language processing, deep learning, and knowledge graphs.