Graph-based Approaches to Text Generation

Vom Fachbereich Informatik
der Technischen Universität Darmstadt
genehmigte

Dissertation

zur Erlangung des akademischen Grades Dr.-Ing.

vorgelegt von
Leonardo Filipe Rodrigues Ribeiro
geboren in Ipatinga, Brazil

Tag der Einreichung: 11. April 2022
Tag der Disputation: 02. Juni 2022

Referenten: Prof. Dr. Iryna Gurevych, Darmstadt, Germany
Dr. Claire Gardent, Nancy, France
Prof. Dr. Yue Zhang, Hangzhou, China

Darmstadt 2022
D17

Ribeiro, Leonardo F. R.: Graph-based Approaches to Text Generation
Darmstadt, Technische Universität Darmstadt
Year thesis published in TUprints: 2022
Day of the viva voce: 02. June 2022
Please cite this document as
URN: urn:nbn:de:tuda-tuprints-214985
URL: https://tuprints.ulb.tu-darmstadt.de/id/eprint/21498

This document is provided by TUprints,
E-Publishing-Service of the TU Darmstadt
http://tuprints.ulb.tu-darmstadt.de
mailto:tuprints@ulb.tu-darmstadt.de

Ehrenwörtliche Erklärung[1]

Hiermit erkläre ich, die vorgelegte Arbeit zur Erlangung des akademischen Grades "Dr.-Ing." mit dem Titel "Graph-based Approaches to Text Generation" selbständig und ausschließlich unter Verwendung der angegebenen Hilfsmittel erstellt zu haben. Ich habe bisher noch keinen Promotionsversuch unternommen.

Darmstadt, den 11. April 2022

Leonardo Filipe Rodrigues Ribeiro

[1] Gemäß §9 Abs. 1 der Promotionsordnung der TU Darmstadt

Wissenschaftlicher Werdegang des Verfassers[2]

01/09 – 08/14	Bachelor of Science (B.Sc.) in Computer Engineering, Centro Federal de Educação Tecnológica de Minas Gerais.
11/15 – 08/18	Master of Science (M.Sc.) in Computer and Systems Engineering, Universidade Federal do Rio de Janeiro.
11/18 – heute	Doktorand, Ubiquitous Knowledge Processing (UKP-Lab), Technische Universität Darmstadt.

[2] Gemäß §8 Abs. 1 lit. a der Promotionsordnung der TU Darmstadt

Abstract

Deep Learning advances have enabled more fluent and flexible text generation. However, while these neural generative approaches were initially successful in tasks such as machine translation, they face problems – such as unfaithfulness to the source, repetition and incoherence – when applied to generation tasks where the input is structured data, such as graphs. Generating text from graph-based data, including Abstract Meaning Representation (AMR) or Knowledge Graphs (KG), is a challenging task due to the inherent difficulty of properly encoding the input graph while maintaining its original semantic structure. Previous work requires linearizing the input graph, which makes it complicated to properly capture the graph structure since the linearized representation weakens structural information by diluting the explicit connectivity, particularly when the graph structure is complex.

This thesis makes an attempt to tackle these issues focusing on two major challenges: first, the creation and improvement of neural text generation systems that can better operate when consuming graph-based input data. Second, we examine text-to-text pretrained language models for graph-to-text generation, including multilingual generation, and present possible methods to adapt these models pretrained on natural language to graph-structured data.

In the first part of this thesis, we investigate how to directly exploit graph structures for text generation. We develop novel graph-to-text methods with the capability of incorporating the input graph structure into the learned representations, enhancing the quality of the generated text. For AMR-to-text generation, we present a dual encoder, which incorporates different graph neural network methods, to capture complementary perspectives of the AMR graph. Next, we propose a new KG-to-text framework that learns richer contextualized node embeddings, combining global and local node contexts. We thus introduce a parameter-efficient mechanism for inserting the node connections into the Transformer architecture operating with shortest path lengths between nodes, showing strong performance while using considerably fewer parameters.

The second part of this thesis focuses on pretrained language models for text generation from graph-based input data. We first examine how encoder-decoder text-to-text pretrained language models perform in various graph-to-text tasks and propose different task-adaptive pretraining strategies for improving their downstream performance. We then propose a novel structure-aware adapter method that allows to directly inject the input graph structure into pretrained models, without updating their parameters and reducing their reliance on specific representations of the graph structure. Finally, we investigate multilingual text generation from AMR structures, developing approaches that can operate in languages beyond English.

Acknowledgments

Ph.D. has been a life-changing experience for me, and I would like to take this opportunity to thank everyone who contributed to making it so memorable.

Firstly, I would like to thank Iryna Gurevych for giving me the opportunity to pursue a Ph.D. under her supervision, for her outstanding feedback, and for always supporting me throughout this journey. When we first met, I didn't know much about NLP – so the fact that Iryna was willing to give me a chance is incredible. I would like to thank Claire Gardent and Yue Zhang for their precious guidance and inputs, generous time, and for agreeing to be reviewers of this thesis. I would further like to thank Ido Dagan for receiving me to his lab in Israel at the beginning of my Ph.D. and for offering valuable feedback on my work on various occasions.

I'm very thankful to all my colleagues from the AIPHES research group and from the UKP Lab for the constructive and helpful feedback that I received during various talks and discussions. In particular, I would like to thank my fellow Ph.D. students (random order), Avinesh P.V.S, Prasetya Ajie Utama, Wei Zhao, Fabrizio Ventola, Tilman Beck, Ji-Ung Lee, Yevgeniy Puzikov, Tim Baumgärtner, Shweta Mahajan, Markus Zopf, Tobias Falke, Jonas Pffeifer, Andreas Hanselowski, Michael Bugert, Aissatou Diallo and Federico López for the deep and insightful conversations during our formal and informal meetings. Thanks for all the support and friendships throughout the last years. I also thank Sue Messenger for all the support given when I arrived in Germany, for the great conversations, and for helping with the bureaucracies of daily life. My sincere appreciation goes to DFG for kindly financing this Ph.D. study through the "Adaptive Preparation of Information from Heterogeneous Sources" (AIPHES, GRK 1994/1) research training group.

I owe a great debt of gratitude to Daniel Figueiredo at the Universidade Federal do Rio de Janeiro, who provided me with a comprehensive and solid education at the very beginning of my academic path, throughout my studies in 2015-2018. In this way, he has inspired me and supported me to become a better scientist, teacher, and leader.

I would like to express my appreciation to Markus Dreyer for hosting me for an internship with his team at Amazon Alexa AI in the summer of 2021. Thanks to Markus Dreyer, Mengwen Liu, Sandeep Atluri and Mohit Bansal for their inspirational and supportive mentorship and for making this a wonderful internship.

Many thanks also go to my family and friends for their support, understanding and simply for sharing their lives with me. Thanks to my friends in Germany: Tiago and Giulia, for the fun, travels and enthusiastic discussions. Agnes and Felix, for all the support, affection and many more. Brazillian friends in Darmstadt for the fun times, weekend barbecues, and dinners. Lola, for each of the peaceful and thoughtful walks in the park and for always being able to make me smile. Thank you all for being family away from home and making my stay in Germany a memorable one.

I would like to thank my parents, José Ribeiro and Andreisa Rodrigues, for their

love and support, hard work for the family, and for raising me to be curious at all times. Even when I decided to go to the other side of the world for many years, they have been nothing but supportive. My achievement is undeniably due to their parenting.

Lastly, and more importantly, I would like to express my deepest gratitude to Nathália Araújo, my partner in life and most trusted friend. Thank you for coming with me to Germany and coping with all the busy weekends, long travels and white nights before deadlines. I'm so grateful for all your endless love, support, and patience along this journey. Thank you for enduring my absence, and for always being or having been there for me. This work is dedicated to you.

Sincerely,
Leonardo Ribeiro

Contents

I Synopsis **1**

Publications and My Contributions **2**

1 Introduction **6**
1.1 Thesis Outline . . . 8

2 Text Generation from Structured Data **11**
2.1 Text Generation . . . 11
2.2 Generating Text from Structures . . . 13
2.2.1 Abstract Meaning Representation to Text . . . 15
2.2.2 Knowledge Graphs to Text . . . 16

3 Graph-to-Text Generation **19**
3.1 Task Definition . . . 19
3.2 Encoder-Decoder Model . . . 20
3.2.1 Transformers . . . 20
3.2.2 Linearized Graph Representation . . . 21
3.3 Encoding the Graph Structure . . . 22
3.3.1 Graph Neural Networks . . . 22
3.3.2 Graph-to-text Architecture . . . 24
3.3.3 Our Contributions . . . 24
3.4 Graph-to-Text Generation with Pretrained Language Models . . . 26
3.4.1 Pretrained Language Models . . . 26
3.4.2 Our Contributions . . . 28

II Publications **31**

4 Enhancing AMR-to-Text Generation with Dual Graph Representations **32**
4.1 Introduction . . . 33
4.2 Related Work . . . 34
4.3 Graph-to-Sequence Model . . . 34
4.3.1 Graph Preparation . . . 34
4.3.2 Dual Graph Encoder . . . 35
4.3.3 Graph Neural Networks . . . 36
4.3.4 Decoder . . . 37
4.4 Data . . . 37
4.5 Experiments and Discussion . . . 37
4.5.1 Implementation Details . . . 37
4.5.2 Results . . . 38
4.5.3 Additional Training Data . . . 39

4.5.4 Ablation Study . . . 39
4.5.5 Impact of Graph Size, Arity and Sentence Length . . . 39
4.5.6 Semantic Equivalence . . . 40
4.5.7 Human Evaluation . . . 40
4.5.8 Semantic Adequacy . . . 41
4.6 Conclusion . . . 42
4.7 Appendix . . . 45

5 Modeling Global and Local Node Contexts for Text Generation from Knowledge Graphs 46
5.1 Introduction . . . 47
5.2 Related Work . . . 48
5.2.1 AMR-to-Text Generation . . . 48
5.2.2 KG-to-Text Generation . . . 49
5.3 Graph-to-Text Model . . . 49
5.3.1 Graph Preparation . . . 49
5.3.2 Graph Neural Networks . . . 50
5.3.3 Global Graph Encoder . . . 50
5.3.4 Local Graph Encoder . . . 51
5.3.5 Combining Global and Local Encodings . . . 51
5.3.6 Decoder and Training . . . 52
5.4 Data and Preprocessing . . . 52
5.5 Experiments . . . 53
5.5.1 Results on AGENDA . . . 53
5.5.2 Results on WebNLG . . . 54
5.5.3 Development Experiments . . . 55
5.5.4 Ablation Study . . . 55
5.5.5 Impact of the Graph Structure and Output Length . . . 56
5.5.6 Human Evaluation . . . 57
5.5.7 Additional Experiments . . . 58
5.6 Conclusion . . . 59

6 Modeling Graph Structure via Relative Position for Text Generation from Knowledge Graphs 63
6.1 Introduction . . . 64
6.2 Related Work . . . 65
6.3 The Graformer Model . . . 65
6.3.1 Graph Data Structure . . . 65
6.3.2 Graformer encoder . . . 66
6.3.3 Self-attention for Text and Graphs with Relative Position Embeddings . . . 67
6.3.4 Graformer Decoder . . . 68
6.3.5 Training . . . 68
6.4 Experiment . . . 68
6.4.1 Datasets . . . 68
6.4.2 Data Preprocessing . . . 69
6.4.3 Hyperparameters and Training Details . . . 69

6.4.4 Epoch Curriculum . . . 69
6.5 Results and Discussion . . . 69
6.5.1 Overall Performance . . . 69
6.5.2 Performance on Different Types of Graphs . . . 70
6.5.3 Ablation Study . . . 71
6.6 Learned Graph Structure . . . 71
6.7 Conclusion . . . 71
6.8 Appendix . . . 74

7 Investigating Pretrained Language Models for Graph-to-Text Generation 76
7.1 Introduction . . . 77
7.2 Related Work . . . 78
7.2.1 Graph-to-Text Learning . . . 78
7.2.2 Pretrained Language Models . . . 78
7.3 PLMs for Graph-to-Text Generation . . . 79
7.3.1 Models in This Study . . . 79
7.3.2 Task-specific Adaptation . . . 79
7.4 Datasets . . . 79
7.4.1 Additional Task-specific Data . . . 80
7.5 Experiment . . . 80
7.5.1 Results on AMR-to-Text . . . 81
7.5.2 Results on WebNLG . . . 81
7.5.3 Results on AGENDA . . . 82
7.5.4 Human Evaluation . . . 82
7.5.5 Limiting the Training Data . . . 83
7.6 Influence of the Graph Structure . . . 83
7.6.1 Quantitative Analysis . . . 83
7.6.2 Qualitative Analysis . . . 84
7.7 Conclusion . . . 85
7.8 Appendix . . . 90

8 Structural Adapters in Pretrained Language Models for AMR-to-Text Generation 94
8.1 Introduction . . . 95
8.2 Related Work . . . 96
8.2.1 Fine-tuning for Graph-to-Text Generation . . . 96
8.2.2 Lightweight Fine-tuning . . . 96
8.3 Graph-to-Text Model . . . 97
8.3.1 Encoder-Decoder Architecture . . . 97
8.3.2 Fine-tuning . . . 97
8.3.3 Baseline Adapter . . . 97
8.3.4 Limitation . . . 98
8.4 Structural Adapter . . . 98
8.4.1 Intuition . . . 98
8.4.2 Graph Representation . . . 98
8.4.3 Method . . . 98

8.5 Experiments . . . 99
8.5.1 Main Results . . . 99
8.5.2 Human Evaluation . . . 100
8.5.3 Detailed Discussion . . . 100
8.6 Graph Representation Evaluation . . . 102
8.6.1 Impact of the Graph Representation . . . 102
8.6.2 Robustness to Graph Linearization . . . 102
8.6.3 Graph Properties . . . 103
8.7 Conclusion . . . 103
8.8 Appendix . . . 108

9 Smelting Gold and Silver for Improved Multilingual AMR-to-Text Generation 109
9.1 Introduction . . . 110
9.2 Related Work . . . 111
9.3 Multilingual AMR-to-Text Generation . . . 111
9.3.1 Approach . . . 111
9.3.2 Data . . . 111
9.3.3 Creating Silver Training Data . . . 112
9.4 Experiments . . . 112
9.5 Conclusion . . . 114
9.6 Appendix . . . 117

III Epilogue 119

10 Conclusion and Future Work 120
10.1 Conclusion . . . 120
10.2 Future Work . . . 121

Bibliography 122

Appendix A Data Handling 138

Part I

Synopsis

Publications and My Contributions

This thesis is based on six scientific publications that I co-authored together with my advisor Iryna Gurevych and many excellent researchers as well as outstanding students: Claire Gardent (French National Center for Scientific Research), Yue Zhang (Westlake University), Martin Schmitt, Philipp Dufter, Hinrich Schütze (LMU Munich) and Jonas Pfeiffer (TU Darmstadt). I thank all co-authors for their significant contributions to these pleasant and successful collaborations. In the following, I detail my own contributions to each publication.

Chapter 4 corresponds to the following publication:

> **Leonardo F. R. Ribeiro**, Claire Gardent and Iryna Gurevych. 2019. Enhancing AMR-to-Text Generation with Dual Graph Representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. pages 3183–3194, Hong Kong, China. Association for Computational Linguistics.

I conceived the original research contributions and performed all implementations. I developed the proposed graph architecture, wrote the initial draft of the article and did most of the subsequent corrections. I regularly discussed the experiments and the paper content with my advisor and Claire Gardent, who assisted me in improving the draft.

Chapter 5 corresponds to the following publication:

> **Leonardo F. R. Ribeiro**, Yue Zhang, Claire Gardent and Iryna Gurevych. 2020. Modeling Global and Local Node Contexts for Text Generation from Knowledge Graphs. In *Transactions of the Association for Computational Linguistics (TACL)*, 8:589–604.

I devised the original research contributions and performed all experiments and evaluations. I proposed the combination of local and global graph encodings and conducted analyses on different datasets. I wrote the first draft of the paper and performed the majority of the revisions. I discussed this work regularly with my advisor, Yue Zhang and Claire Gardent, who helped me improve it.

Chapter 6 corresponds to the following publication:

> Martin Schmitt, **Leonardo F. R. Ribeiro**, Philipp Dufter, Iryna Gurevych, Hinrich Schütze. 2021. Modeling Graph Structure via Relative Position for Text Generation from Knowledge Graphs. In *Proceedings of the TextGraphs-15 Workshop: Graph-based Methods for Natural Language Processing*, pages 10–21, 2021 Annual Conference of the North American Chapter of the Association for Computational Linguistics. Mexico City, Mexico. Association for Computational Linguistics.

I contributed significantly regarding the preprocessing of the input graphs from different datasets (section 4.2), hyperparameter tuning (section 4.3 and appendix A), and evaluation of the generated outputs. I advised Martin Schmitt with suggestions about model design and architecture. In particular, I assisted Martin Schmittin in designing the experiments to assess model performance on input graphs with different specific properties (section 5.2). I regularly discussed this work with the lead author and assisted in improving the draft.

Chapter 7 corresponds to the following publication:

> **Leonardo F. R. Ribeiro**, Martin Schmitt, Hinrich Schütze and Iryna Gurevych. 2021b. Investigating Pretrained Language Models for Graph-to-Text Generation. In *Proceedings of the 3rd Workshop on NLP for ConvAI*. pages 211–227, 2021 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP), Online. Association for Computational Linguistics.

I proposed the initial research idea and preliminary investigations. With Martin Schmitt's assistance, I performed all model implementations, executions, and evaluations. I wrote the initial paper draft and did most of the subsequent corrections and experiments. I regularly discussed this work with the coauthors and we improved the draft together.

Chapter 8 corresponds to the following publication:

> **Leonardo F. R. Ribeiro**, Yue Zhang and Iryna Gurevych. 2021c. Structural Adapters in Pretrained Language Models for AMR-to-text Generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4269–4282, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

I devised the original research contributions and performed all experiments and evaluations. I wrote the initial paper draft and executed the majority of the revisions. I discussed this work regularly with my advisor and Yue Zhang, who helped me improve it.

Chapter 9 corresponds to the following publication:

> **Leonardo F. R. Ribeiro**, Jonas Pfeiffer, Yue Zhang and Iryna Gurevych. 2021a. Smelting Gold and Silver for Improved Multilingual AMR-to-Text Generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP)*, pages 742–750, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

After an initial brainstorming with Yue Zhang and Jonas Pfeiffer, I conceived the initial research idea and performed all implementations and experiments. I preprocessed all the datasets and trained the models. I wrote the initial draft of the article and did most of the subsequent corrections. I regularly discussed this work with the coauthors and we collaboratively improved the final text and experiments.

During my PhD, I was fortunate to work with great researchers on many topics, some of which did not fit into this thesis. In the interest of completeness, I provide references to these papers:

> Tobias Falke, **Leonardo F. R. Ribeiro**, Prasetya Utama, Ido Dagan and Iryna Gurevych. 2019. Ranking Generated Summaries by Correctness: An Interesting but Challenging Application for Natural Language Inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2214–2220, Florence, Italy. Association for Computational Linguistics.

> Anne Lauscher, Olga Majewska, **Leonardo F. R. Ribeiro**, Iryna Gurevych, Nikolai Rozanov, Goran Glavas. 2020. Common Sense or World Knowledge? Investigating Adapter-Based Knowledge Injection into Pretrained Transformers. In *Proceedings of Deep Learning Inside Out (DeeLIO): The First Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 43–49, Online. Association for Computational Linguistics.

> Mohsen Mesgar, **Leonardo F. R. Ribeiro**, Iryna Gurevych. 2021. A Neural Graph-based Local Coherence Model. In *Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2021. Association for Computational Linguistics*, pages 2316–2321, Punta Cana, Dominican Republic. Association for Computational Linguistics.

> Tim Baumgärtner, Kexin Wang, Rachneet Sachdeva, Max Eichler, Gregor Geigle, Clifton Poth, Hannah Sterz, Haritz Puerto, **Leonardo F. R. Ribeiro**, Jonas Pfeiffer, Nils Reimers, Gözde Gül Şahin, Iryna Gurevych. 2022. UKP-SQUARE: An Online Platform for Question Answering Research. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations.* Dublin, Ireland. Association for Computational Linguistics.

Leonardo F. R. Ribeiro, Mengwen Liu, Iryna Gurevych, Markus Dreyer and Mohit Bansal. 2022. FactGraph: Evaluating Factuality in Summarization with Semantic Graph Representations. In *2022 Annual Conference of the North American Chapter of the Association for Computational Linguistics. Seattle, United States. Association for Computational Linguistics.*

The experimental source code of our work related to the contributions listed above is publicly available at https://github.com/ukplab. Details on our strategy to data handling are given in Appendix A.

Chapter 1

Introduction

"Language is the blood of the soul into which thoughts run and out of which they grow."

— Oliver W. Holmes

The ongoing production and consumption of data are among the most evident characteristics of the information age. Much of this information is communicated in the form of natural language. For example, people communicate through instant messaging, seek information using personalized voice assistants, write emails with the help of an autocomposer and use online translation systems to read a foreign website. These applications and a variety of other systems that people interact with on a daily basis, including chatbots and question answering, largely rely on *text generation*.

Text generation focuses on producing realistic and human readable text from input data. Examples of such inputs include documents, multilingual sources, modalities such as images and videos and structured knowledge presented in databases. The generated text should communicate the information encoded in the data to the reader in a fluent and accurate way. This is a difficult task since the fluency of a text accounts for criteria such as grammar, spelling, choice of words, and style, whereas an accurate text must faithfully represent the information conveyed in the input data.

The resurgence of deep learning has considerably advanced this field, mostly through the use of massive datasets to train neural network architectures, greatly improving the fluency of the machine-generated text. In particular, the paradigm of pretrained language models is generally beneficial for downstream text generation tasks. In this paradigm, the basic idea is to first pretrain the model on large-scale unsupervised corpora and then fine-tune this model in downstream supervised tasks. However, existing neural text generation systems face challenges, suffering from multiple limitations from producing text that is not faithful to the input to supporting only English and neglecting other languages.

Many generation systems receive as input *structured data*. In this scenario, structures (e.g., tables or knowledge bases) can be read or listened to, as when a gadget shows sports reports or weather forecasts, or when a voice assistant responds to a user query. An important type of structured input are *graphs*, which can precisely represent relationships in the input. Graphs are a ubiquitous data structure and a universal language for describing a collection of items (nodes) and their pairwise relationships (edges).

In Natural Language Processing (NLP), graph-based structures such as knowledge graphs and semantic representations are suitable for representing and storing knowledge in a convenient and canonical way. In many situations, we want to make use of these forms of structured knowledge and verbalize such information into natural language, a process known as *graph-to-text generation*. Consider, for instance, question answering applications, where answering questions often requires verbalizing in natural language facts present in a knowledge subgraph.

While the field of text generation has seen rapid progress, much research on graph-to-text generation has been directed towards *sequence-to-sequence* models that are broadly successful in many other applications such as neural machine translation. However, a severe limitation of sequence-to-sequence models is that they require linearization of input graphs, which adds to the challenge of representing the graph structure. It can be difficult for a sequential encoder to automatically induce the original graph connections from a linearized form since the connections between nodes are not explicitly considered. The linearized representation weakens structural information in the original graph by diluting the explicit connectivity, especially when the graph input is large, containing several entities and relations. Moreover, while texts are inherently sequential, graphs do not contain a fixed node ordering or reference point, and linearization techniques should not impact the node representations generated by the graph encoder.

Motivated by these two facts — the usefulness of structured graph inputs for text generation and the insufficient manner of representing input graphs for this task — the goal of this thesis is to develop adequately neural encoding mechanisms for input graphs in order to generate fluent and accurate text, advancing text generation based on graph-based data. More specifically, the research presented in this thesis is guided by the following three high-level research questions:

- How to preserve the semantics of the input graph structure (e.g., relative positions of the nodes in the graph and their relations) and suitably encode it into neural encoder-decoder models for improved graph-to-text generation?

- How pretrained language models that have initially been pretrained on natural language perform with graph-based data as input?

- How to adapt such pretrained language models to better process and represent graph inputs in different scenarios such as in multilingual generation?

In order to address these research questions, in this thesis, we study and propose

text generation approaches for various graph-to-text tasks with the capability of incorporating the graph structure into learned representations that better reflect the semantic relations of the nodes in the original input graph. This specialized input encoding allows generating text that is not only more fluent but also conveys the content from the input graph more faithfully.

Transfer learning has become ubiquitous in NLP, and Transformer-based pretrained architectures have considerably outperformed prior state of the art in a wide range of downstream tasks. We investigate large-scale transfer learning using encoder-decoder language models pretrained on text-to-text tasks for graph-to-text generation. We present a study across different graph representations and domains (i.e., meaning representations, Wikipedia KGs, and scientific KGs) and introduce task-adaptive graph-to-text pretraining approaches for pretrained language models. We further adapt those pretrained models for structured data, proposing an adapter-based mechanism that employs layer-wise graph convolution modules to learn representations built upon the graph connectivity over the pretrained encoder without altering its pretrained knowledge. Finally, we explore multilingual text generation from semantic graph representations and propose training strategies that explore combinations of multilingual training data, leading to a stronger multilingual graph-to-text model.

1.1 Thesis Outline

The structure of this thesis follows the same order as the publication record given in *Publications and My Contributions*. Figure 1.1 illustrates the overall thesis structure.

In Chapter 2, we define the basic notions used throughout this thesis and introduce the field of *Text Generation*. We discuss text generation tasks that employ different data formats and modalities as input and review work on text generation from structured data. We then introduce the subtask of *graph-to-text generation* that seeks to generate natural language from graph-based inputs, mainly focusing on Abstract Meaning Representations (AMRs) and Knowledge Graphs (KGs). In Chapter 3, we formally define graph-to-text generation, clarify terminology, and present basic neural methods that are used as building blocks in the thesis. We detail the encoder-decoder architecture and explore neural mechanisms used for learning representations of graph structures, including Transformer and graph neural network models.

In Chapter 4, we propose a dual graph encoder for AMR-to-text generation that encodes complementary perspectives of the AMR graph structure and investigate different graph neural networks techniques for this task. We provide a detailed examination demonstrating that explicitly modeling the input graph connectivity using our dual mechanism allows improving the quality of the generated sentences measured by automatic and human evaluations. In Chapter 5, we present a novel framework for KG-to-text generation that encodes an input graph combining both global and local node contexts, in order to learn richer contextualized node embeddings for improved multi-sentence generation. An extensive evaluation of our

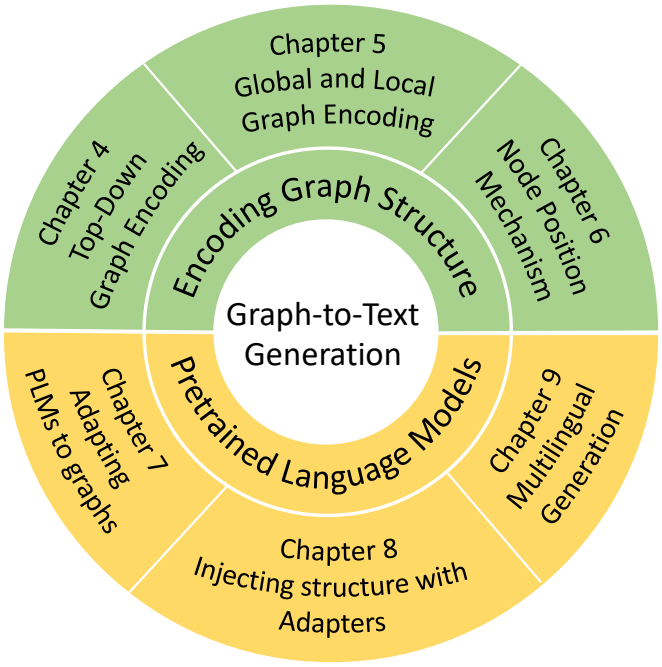


Figure 1.1: The organization of this thesis. We complement this with Chapter 2 containing background information on text generation from structured data, Chapter 3 on text generation from graph-based inputs, and Chapter 10 with conclusions of the thesis and an outlook on possible future work.

framework demonstrates that while the local encoder considers the node neighborhood, the global encoder learns to focus on distant nodes, revealing that both encodings are complementary. In Chapter 6, we introduce a new manner of injecting graph structure in the Transformer architecture for parameter-efficient graph-to-text generation, demonstrating strong performance while using considerably fewer parameters. This approach represents the relation between two nodes as the length of the shortest path between them, capturing diverse node-to-node associations.

In Chapter 7, we then concentrate on the applicability of encoder-decoder pretrained language models for text generation from graph-based data. However, graph-based inputs are different in essence from natural language. As a result, knowledge learned from large-scale pretraining using natural language text intuitively cannot be fully transferred to graph-based data in a fine-tuning phase. We thus present different task-adaptive pretraining strategies for adapting the pretrained model for various graph-to-text tasks. In Chapter 8, we propose to augment pretrained language models with novel structure-aware adapters that allow to directly infuse the input graph structure into the node representations. Adapters only train a newly introduced set of weights at every layer, instead of fully fine-tuning the entire pretrained model, thereby sharing the majority of parameters between tasks. We show that our system is more robust to different graph linearizations and graph attributes, reducing its reliance on specific representations of the graph structure. Finally, in Chapter 9, we develop a multi-task training approach using data augmentation strategies to generate sentences in different languages from the same AMR structure.

Since AMR can be employed as an intermediate meaning representation in diverse NLP tasks, generating text in different languages from this semantic structure can enable multilingual capabilities in such applications.

We conclude in Chapter 10 by summarizing the main research contributions of this thesis and considering future research directions.

Chapter 2

Text Generation from Structured Data

2.1 Text Generation

Text generation, also known as *natural language generation* (Gatt and Krahmer, 2018, Gehrmann et al., 2021), is a branch of Natural Language Processing (NLP) that aims to generate fluent natural language output from a variety of inputs, including texts, tables, meaning representations, knowledge graphs and images, among others. Text generation systems seek to realistically produce text in English and other human languages that communicates ideas to readers in a clear and practical manner, just as individuals express ideas through writing or voice.

Text generation techniques have recently become widely used in a significant number of downstream applications, contributing to the growth of many subfields of natural language generation. In what follows, we present a few examples of text generation tasks where the input is in text form:

- **Machine translation** (e.g., Johnson et al., 2017; Aharoni et al., 2019). Machine translation is the process of automatically converting text from one language to another. These approaches encode a sequence of words in one language (such as English) and decode a sequence in another language (such as Portuguese). Machine translation models are a courier for transmitting knowledge as they allow people to consume content in different languages.

- **Summarization** (e.g., Liu and Lapata, 2019; Zhang et al., 2020). The goal of summarization systems is to provide the key points of a larger text in a concise manner. These systems take a big string of text and condense it into a short and clear version that still maintains the important information. As a result, summarization can help people absorb information more efficiently, whether reading newspaper articles, emails, or extensive reports.

- **Simplification of complex texts** (e.g., Macdonald and Siddharthan, 2016;

Zhang and Lapata, 2017). The purpose of text simplification is to reduce the linguistic complexity of text while maintaining its original meaning and information. Those approaches can improve the readability, understandability and accessibility of textual information, which has important implications for people with low-literacy or limited language skills, such as children and non-native speakers, as well as those with cognitive disabilities.

- **Generation of paraphrases** (e.g., Kauchak and Barzilay, 2006; Xu et al., 2012). Paraphrasing models can generate phrases, sentences, or longer natural language expressions that convey nearly the same information. The production of paraphrases allows for creating more varied and fluent text and enhance applications such as question answering and summarization. In question answering, for example, paraphrases may provide more fluent and concise answers which can also convey more information, whereas in multi-document summarization, generating paraphrases allows information repeated across documents to be compressed and better expressed.

Earlier text generation techniques typically use statistical language models to model conditional probabilities of words given the n-gram context (Brown et al., 1990, 1993; Brown and Frederking, 1995). However, such statistical methods are likely to suffer from the problem of data sparsity, and different smoothing strategies have been proposed to better estimate the occurrence of unobserved terms (Zhai and Lafferty, 2001; Tao et al., 2006). Neural network models have produced remarkable improvements and dominated the mainstream methods in text generation since the emergence and development of deep learning techniques (LeCun et al., 2015). These approaches are mainly based on the *encoder-decoder* neural network architecture (Sutskever et al., 2014) and are often referred to as *sequence-to-sequence models* as they take as input a sequence, and then output a sequence, one element at a time. The encoder maps the input sequence into vector representations that are used by the decoder to generate the target text. The decoder essentially behaves as a language model conditioned on the input sequence and the previously generated tokens. Specifically, the decoder is connected to the encoder through an attention mechanism (Bahdanau et al., 2015) that allows the decoder to automatically focus on parts of the source input that are relevant to predicting the target text. Several architectures based on neural networks have been used for both encoders and decoders, such as Recurrent Neural Networks (Rumelhart et al., 1986; Hopfield, 1982), Long Short-Term Memory models (Hochreiter and Schmidhuber, 1997), Convolutional Neural Networks (Gehring et al., 2017) and recently Transformers (Vaswani et al., 2017). Despite this recent success, multiple issues with neural generation persist, such as factual inconsistencies with respect to the input data and poor performance in out-of-domain, few-shot, and multilingual settings (Celikyilmaz et al., 2021). This thesis explores several aspects of text generation, from developing better encoding models for input data to evaluating proposed methods and multilingual settings.

2.2 Generating Text from Structures

Various text generation systems produce natural language output based on information contained in textual input, such as those used in machine translation, sentence simplification or summarization. However, it is frequently required to produce texts that are not based on existing ones. For instance, automatically generating captions for images or videos is a important part of scene understanding and an emerging interdisciplinary problem (Venugopalan et al., 2015; He and Deng, 2017).

A common input to text generation systems are structured representations (Gardent et al., 2017; Parikh et al., 2020). In this scenario, also known as *data-to-text generation* (Gatt and Krahmer, 2018) or sometimes referred to as *structured input-to-text generation* (Gehrmann et al., 2021; Xie et al., 2022), different types of structured data (e.g., tables, source codes and knowledge graphs) can be verbalized in natural language. For example, structured records such as gaming databases (Wiseman et al., 2017) and dialogue acts expressed using meaning representations (Wen et al., 2015; Juraska et al., 2019) are typical sources for data-to-text models. Another example is automatic question generation (Rus et al., 2011; Rao and Daumé III, 2018) that aims to produce questions from non-textual inputs such as databases and tables, and is an important task for different applications, such as dialogue systems, intelligent tutoring systems, and search interfaces. Practical data-to-text approaches can be found in domains such as finance (Plachouras et al., 2016), weather forecasts (Mei et al., 2016), health care (Portet et al., 2009), election results (Leppänen et al., 2017) and sportscasting news (Chen and Mooney, 2008; van der Lee et al., 2017).

Early works in data-to-text generation were mostly based on rules and templates (McKeown, 1982; Kukich, 1983) and statistical methods (Och et al., 1999; Koehn et al., 2007; Belz, 2008; Konstas and Lapata, 2013), frequently using a pipeline architecture (Reiter and Dale, 2000). Examples of standard components present in this pipeline include (i) *document planning*, which determines what information should be incorporated; (ii) *microplanning*, which defines how to organize and express information; and (iii) *realization*, which transforms abstract representations into a fluent, natural language text.

Currently, most deep learning approaches, using the encoder-decoder paradigm, consolidate all the data-to-text generation classical components into a single end-to-end system, achieving substantial improvements in numerous text generation tasks. The neural encoder-decoder architecture provides a natural and unifying framework for text generation, regardless of the type of input (e.g., text, records, knowledge graphs or meaning representations) producing highly fluent, natural sounding text. Due to its flexibility, these models have been applied to a range of different generation applications that take structured data as input, including:

- **Source code summarization** (e.g., Wan et al., 2018, Ahmad et al., 2020). Source code summarization refers to the task of creating understandable text summaries that describe the functionality of a program and can facilitate software development, documentation and maintenance. A natural language description of source code promotes program comprehension by greatly reducing

the efforts of the developer.

- **Table-to-text generation** (e.g., Liu et al., 2018; Parikh et al., 2020). The goal of table-to-text generation is to automatically generate natural-language descriptions from data recorded in tables. This task assists humans in easily comprehending knowledge elements in tables and their relationships. There have been several practical applications in this field, e.g., generating biographies based on Wikipedia infoboxes or tables, sport news writing, and medical-record description generation.

- **SQL-to-text generation** (e.g., Xu et al., 2018b; Shu et al., 2021). SQL (structured query language) is an essential mechanism to access databases. However, SQL is not easy to understand for non-expert users. SQL-to-text generation aims to convert a structured SQL program into a text, enabling people to understand complex SQLs quickly by reading the corresponding natural language description. It can support automatic SQL comment generation and help to comprehend elaborate SQL queries that are automatically generated.

- **Generative question answering** (e.g., Fan et al., 2019; Xie et al., 2022). Question answering aims to build systems that automatically answer questions posed by users in natural language and is typically used as part of chatbots, social media and speech-enabled apps, and search engines. In generative question answering, many systems rely on structured data to produce natural language passages that answer complex questions. This structured information can reside in heterogeneous databases or knowledge bases and is important for answering multi-hop questions that require reasoning and information integration.

In this thesis, we concentrate on *graph-to-text generation* (Song et al., 2018; Koncel-Kedziorski et al., 2019), a subtask of data-to-text generation that seeks to produce fluent and consistent text from *graph-based inputs*. Generally, these inputs are composed of collections of concepts or entities and relations among them. Structured representations in the form of graphs can store knowledge in a machine-readable format and provide means for information to be collected, organized, searched and easily utilized. For example, information on Wikipedia can be expressed as knowledge graphs using structured knowledge bases such as Wikidata (Vrandečić and Krötzsch, 2014)[1] and DBPedia (Lehmann et al., 2015),[2] in addition to free-text form.

Syntactic or semantic formalisms are another type of graph-based input to text generation systems. Those formalisms can be used as intermediate meaning representations that determine the information to be generated into a natural language text. For instance, dependency graphs are semantic inputs that represent a target sentence using predicate-argument structures and are used in surface realization

[1] https://www.wikidata.org
[2] https://www.dbpedia.org

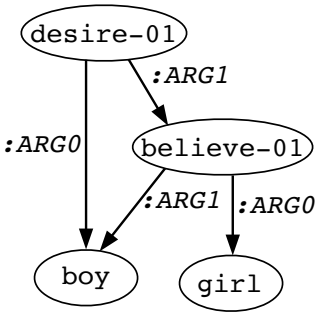


Figure 2.1: Abstract Meaning Representation (AMR) for the sentence *The boy desires the girl to believe him.*

tasks (Belz et al., 2011; Mille et al., 2018). Another form of structured semantic representation is Abstract Meaning Representation (AMR, Banarescu et al., 2013), which is a graph-based representation that captures semantics at a higher level of abstraction than plain text. AMR aims towards a logical, consistent, and less syntactic representation. Other structured representations that have been utilized as inputs for text generation includes first-order logic (Gerdemann and Hinrichs, 1990) and minimal recursion semantic (Hajdik et al., 2019) representations, discourse structures (Basile and Bos, 2011), lambda calculus expressions (Lu and Ng, 2011), and other grammar formalisms (Cahill and van Genabith, 2006; White et al., 2007).

In this thesis, we focus on generating text from two types of graph-based structured inputs, namely, Abstract Meaning Representation (AMR) and Knowledge Graphs (KG), which we detail in the following sections.

2.2.1 Abstract Meaning Representation to Text

Abstract Meaning Representation (AMR, Banarescu et al., 2013) is a linguistically-grounded semantic formalism that represents the meaning of a sentence as a rooted directed graph, where nodes are concepts and edges are semantic relations. AMR aims to abstract away from syntactic idiosyncrasies, and is not intended to convey all of the information in a natural language sentence, eliminating, for example, factors such as tense, articles and plurality.

For instance, consider the following five sentence constructions:[3]

- The boy desires the girl to believe him.
- The boy desires to be believed by the girl.
- The boy has a desire to be believed by the girl.
- The boy's desire is for the girl to believe him.

[3] This example is adapted from the excellent AMR guidelines: https://github.com/amrisi/amr-guidelines.

- The boy is desirous of the girl believing him.

These sentences can be represented by the same AMR, shown in a graphical form in Figure 2.1 and using the PENMAN[4] notation:

```
(d / desire-01
    :ARG0 (b / boy)
    :ARG1 (b2 / believe-01
        :ARG0 (g / girl)
        :ARG1 b))
```

This graph-based structure condenses the key conceptual information from a number of different sentences into a single compact representation of semantic meaning. As a result, AMR is able to distinguish semantic meaning from the sentence's surface appearance.

Since AMR represents a sentence in a form that abstracts from morphological and syntactic variability, its use is advantageous in many semantic NLP tasks, including text summarization (Hardy and Vlachos, 2018; Dohare et al., 2018; Lee et al., 2021), machine translation (Song et al., 2019), spoken language understanding (Damonte et al., 2019), commonsense reasoning (Lim et al., 2020), and question answering (Kapanipathi et al., 2021; Bornea et al., 2021). While initial AMR research has focused primarily on English, recent work shows that it is also possible to use AMR as a semantic representation for sentences written in other languages, such as Brazilian Portuguese, Chinese, German, Italian and Spanish (Damonte and Cohen, 2018; Migueles-Abraira et al., 2018; Sobrevilla Cabezudo and Pardo, 2019).

The purpose of AMR-to-text generation is to produce a text which verbalizes the meaning encoded by an input AMR graph. This is a challenging task as capturing the complex structural information stored in graph-based data is not trivial, demanding the usage of neural models adapted to encode graph structures rather than standard sequence-to-sequence approaches. Additionally, AMR is not meant to thoroughly represent all information within a sentence (e.g., it does not capture information about verb tenses). This contrasts with text generation tasks such as paraphrase generation, where the input contains the complete information required to generate the new sentence. As a result, AMR-to-text generation is essentially distinct from other text generation tasks since the models are required to additionally complete the missing details when generating the text based on the graph-based input.

2.2.2 Knowledge Graphs to Text

There is an abundance of highly accurate knowledge in different forms of structured data arranged in knowledge bases and knowledge graphs (KGs). KGs are often used to describe factual knowledge in the form of relations between entities, as presented in Figure 2.2. Commonly, KG data can be represented in the triple format

[4] PENMAN notation is a serialization format for directed, rooted graphs used to represent semantic dependencies, most notably in the AMR framework.

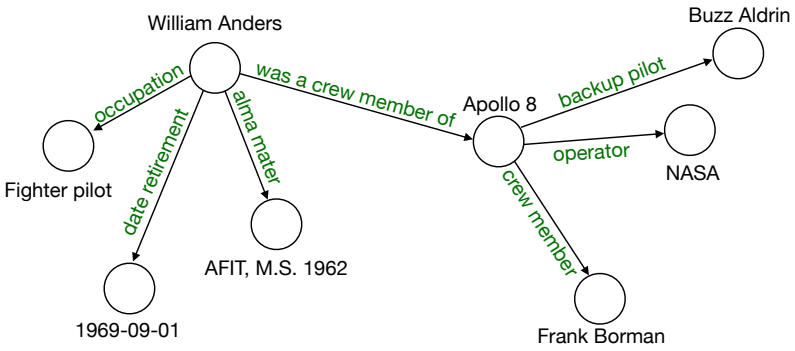


Figure 2.2: Example of a knowledge graph describing relations between entities. The following text verbalizes the KG information: *William Anders received a M.S. from his Alma Mater, AFIT in 1962, was a fighter pilot on Apollo 8 and retired on Sept 1st 1969. Buzz Aldrin was a backup pilot on the mission with William Anders and Frank Borman as crewmen.* Adapted from the WebNLG dataset.[5]

(e.g., using the resource description framework, RDF), where each triple contains a subject, a relation, and an object. Different KG domains and forms are present in the literature, such as Wikipedia-based KGs (Vougiouklis et al., 2018), scientific KGs (Koncel-Kedziorski et al., 2019) and biomedical and clinical KGs (Wishart et al., 2017). As an example, KG plays an important role in efficiently representing document content, especially when the text length is large, benefiting summarization (Huang et al., 2020). In the medical domain, KGs extracted from clinical data are represented in the RDF format and used as input to text generation models that produce patient records in natural language (Bontcheva and Wilks, 2004). Moreover, KGs can be developed in a multilingual setting, allowing the implementation of knowledge-intensive applications supporting multiple languages (Speer et al., 2017; Shimorina et al., 2019; Castro Ferreira et al., 2020). In recent years, crowdsourcing platforms and information extraction systems (IE) have been used extensively to create labeled pairs of KG and their descriptive text (Lebret et al., 2016; Gardent et al., 2017).

Generating text from KGs is an important task for effectively disseminating knowledge and is part of different NLP applications, including question answering approaches present in voice assistants, chatbots or search engines. When a user requests information about an actor, for example, the system must not only extract a sub-graph relating to that individual, but also express these triples fluently in natural language text. In contrast to AMR-to-text generation, in the KG-to-text scenario, the natural language output can consist of multi-sentence descriptions of KG entities, as shown in Figure 2.2. Consequently, appropriate words and multi-sentence structures must be correctly chosen to produce a coherent text that describes KB entities and their relations. Increases in model capacity and data availability have enabled the generation of mostly grammatical and fluent sentence-level natural language text. However, it remains an open task how to accurately generate multiple sentences linked to a topic that exhibit general coherence and

[5] https://webnlg-challenge.loria.fr/

discourse-relatedness. Traditional techniques divide the KG-to-text generation task into several micro-tasks, including discourse ordering, sentence structuring, lexicalization, and generating reference expressions (Castro Ferreira et al., 2019).

Chapter 3

Graph-to-Text Generation

Graph-to-text generation aims to convert a graph-based input into text that conveys the content shown in the graph. To accomplish this task, common approaches linearize the input graph into a string and train a sequence-to-sequence model from scratch. Alternatively, neural architectures that seek to explicitly model the graph structure, i.e., which nodes are connected to each other, can be used to capture the input graph's structural and semantic properties. In this chapter, we formally define the graph-to-text generation task and present some of the basic components used as part of our proposed approaches. We then summarize our main contributions and findings which are detailed in the following chapters.

3.1 Task Definition

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$ denote a directed graph with a node set $\mathcal{V}$ and labeled edges $(u, r, v) \in \mathcal{E}$, where $u, v \in \mathcal{V}$ and $r \in \mathcal{R}$ is a relation type. Examples of such graph are AMRs (Figure 2.1) and KGs (Figure 2.2).

In a graph-to-text setting, we transduce the input graph $\mathcal{G}$ to its corresponding surface realization $y = \langle y_1, \ldots, y_M \rangle$ via a parameterized probabilistic model $p_\theta(\cdot)$. The specific semantics of $\mathcal{G}$ varies depending on the task at hand. In linearized approaches specifically, the graph $\mathcal{G}$ is first mapped to a sequence by way of a linearization function $x = \text{LIN}(\mathcal{G})$, and $p_\theta(\cdot)$ is an autoregressive decoder or sequence-to-sequence model. The model is then training using the following likelihood objective:

$$p_\theta(y \mid \mathcal{G}) = \prod_{i=1}^{M} p_\theta(y_i \mid x, y_{1:i-1}). \tag{3.1}$$

Usually, beam search (Shao et al., 2017) is employed to generate the target text during the decoding stage.

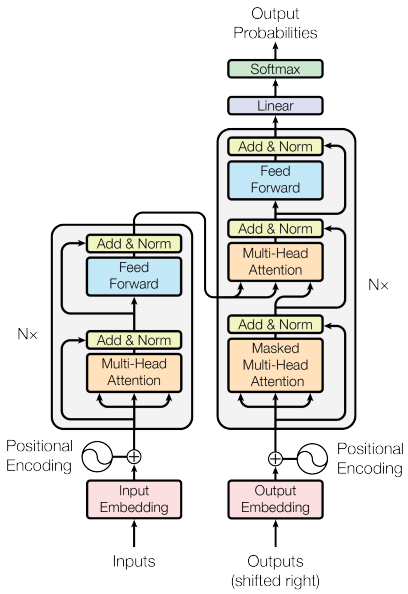


Figure 3.1: The Transformer architecture. Nx denotes a stack of N layers. Figure taken from Vaswani et al. (2017).

3.2 Encoder-Decoder Model

Sequence-to-sequence models based on the encoder-decoder architecture are suitable for text-to-text generation and can also be adapted to graph-to-text generation. In sequence-to-sequence models, the encoder reads the input and generates a representation of it, which the decoder then utilizes to generate the output text. Here, we illustrate a sequence-to-sequence model using the Transformer architecture (Vaswani et al., 2017), where both encoder and decoder take a sequential form.

3.2.1 Transformers

Transformer maps a sequence of input tokens $x = \langle x_1, ..., x_N \rangle$ to a sequence of output tokens $y = \langle y_1, \ldots, y_M \rangle$. The Transformer consists of encoder and decoder modules, each of which are made up of stacks of transformer layers, as shown in Figure 3.1. In the Transformer architecture, the initial embeddings of the tokens contain positional encodings, which inject information about the relative or absolute position of the tokens in the sequence.

Each encoder layer l has two sublayers: a self-attention layer followed by a position-wise feed forward layer. The self-attention mechanism employs K attention heads. These multiple heads allow the model to jointly attend to information from distinct representation subspaces at different positions. Each head takes the layer input representation $\mathbf{h}_i \in \mathbb{R}^d$ of each $x_i \in x$, to calculate the scaled dot-product attention:

$$\mathbf{z}_i = \sum_{j=1}^{N} \alpha_{ij} \mathbf{W}_v \mathbf{h}_i \quad . \tag{3.2}$$

Specifically, the attention weight α_{ij} of each element x_j is computed using a

softmax function:

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{N} exp(e_{ik})} \tag{3.3}$$

where:

$$e_{ij} = \frac{(\mathbf{W}_q \mathbf{h}_i)^T (\mathbf{W}_k \mathbf{h}_j)}{\sqrt{d_z}} \tag{3.4}$$

is an alignment function which measures how well the input elements x_i and x_j match. $\mathbf{W}_v, \mathbf{W}_q, \mathbf{W}_k \in \mathbb{R}^{d_z \times d_x}$ are parameters to be learned. The results from all the attention heads are concatenated together and a parameterized linear transformation is applied to get the output of the self-attention sublayer:

$$\hat{\mathbf{z}}_i = \mathbf{W}_o \Big|\Big|_{k=1}^{K} \mathbf{z}_i^{(k)} \ . \tag{3.5}$$

Finally, layer normalization (LayerNorm, Ba et al., 2016) and a fully connected feed-forward network (FFN) are employed:

$$\tilde{\mathbf{z}}_i = \text{LayerNorm}(\hat{\mathbf{z}}_i + \mathbf{h}_i) \,, \tag{3.6}$$

$$\mathbf{y}_i = \text{LayerNorm}(\text{FFN}(\tilde{\mathbf{z}}_i) \ + \ \tilde{\mathbf{z}}_i) \,. \tag{3.7}$$

As shown in Figure 3.1, each decoder layer has an additional multi-head attention component that links it to the encoder. In particular, each time step t is updated by performing multi-head attentions over the output of the encoder and over previously-generated tokens (token embeddings).

3.2.2 Linearized Graph Representation

One of the main challenges of graph-to-text generation is how to represent and encode the input graph structure. Previous efforts (Konstas et al., 2017; Castro Ferreira et al., 2019; Hoyle et al., 2021) first linearize the input graph with depth-first traversal, before feeding it into the sequence encoder. However, linearizing the graph may lose crucial structural information. For example, originally closely-related nodes (such as parents and children) can be far away after the linearization, especially when the graph is very large. Moreover, deep learning models for graphs should be permutation invariant to node order in the linearization. However, different graph traversal algorithms lead to different linearizations, increasing data variation and additionally introducing ambiguity. It is also worth noting that during the input encoding of the Transformer, the self-attention mechanism computes a token representation based on all tokens in the linearization. In this way, when the input tokens correspond to a linearized graph $x = \text{LIN}(\mathcal{G})$, the graph connectivity is diluted and the model needs to infer about the graph connections. In addition, the positional encoding of the token embeddings makes the model more dependent on the linearization strategy used.

Despite the drawbacks presented, sequential encoders are able to achieve competitive results when large-scale training data is available. For instance, Konstas et al. (2017) employ a sequential approach to encode millions of automatically parsed AMR graphs paired with sentences for text generation, achieving strong performance. These results show the capacity of neural encoders for learning good representations of graph-based data when plenty of data is provided. Nonetheless, there is much room for improvement due to the explicit loss of structural information. We discuss in the next section neural approaches that make explicitly use of the node associations and graph properties in order to learn better representations.

3.3 Encoding the Graph Structure

The purpose of graph neural encoding is to generate representations of nodes that explicitly depend on the structure of the graph. This encoding approach effectively exploits the connectivity between nodes, and the composition of the graph has a substantial impact on the node representation. In this thesis, we focus on an important neural architecture used to encode graph structures: graph neural networks.

3.3.1 Graph Neural Networks

Graph neural networks (GNNs) are a class of neural models suitable for processing graph-structured data. The key idea is computing representations of nodes based on their local neighborhood (context) and in feature information (from the nodes and edges).[1] Gori et al. (2005), who coined the term graph neural network, propose many of the core ideas found in the GNN methods and are generally credited with developing the first GNN model.

GNNs use an information propagation strategy in which each node's representation is updated iteratively. During an iteration, information is gathered from each node's neighbors and then used to compute the node representation. In the course of these iterations, each node embedding incorporates more and more information from further parts of the graph. Whereas the node order impacts a sequence encoder, GNNs are invariant to the permutations of the nodes and edges, which better captures the graph connectivity.

Specifically, in each message-passing iteration in a GNN, the representation $\mathbf{h}_v \in \mathbb{R}^d$ corresponding to a node $u \in \mathcal{V}$ is updated according to information aggregated from both its context node neighbors and edge features. Formally, the l-th layer aggregates the representations of v's context nodes:

$$\mathbf{h}_{\mathcal{N}(v)}^{(l)} = \text{AGGR}^{(l)}\left(\left\{\left(\mathbf{h}_u^{(l-1)}, r_{uv}\right) : u \in \mathcal{N}(v)\right\}\right),$$

where $\text{AGGR}^{(l)}(\cdot)$ is an aggregation function, shared by all nodes on the l-th iteration/layer. $r_{uv} \in \mathcal{R}$ represents the relation between u and v, and $\mathcal{N}(v)$ is the

[1] Note, however, that there are GNN models that compute node representations using information from nodes other than those present in the original local neighborhood.

immediate neighborhood of v. $\mathbf{h}_{\mathcal{N}(v)}^{(l)}$ is the aggregated context representation of $\mathcal{N}(v)$ at layer l. $\mathbf{h}_{\mathcal{N}(v)}^{(l)}$ is used to update the representation of v:

$$\mathbf{h}_v^{(l)} = \text{COMBINE}^{(l)}\left(\mathbf{h}_v^{(l-1)}, \mathbf{h}_{\mathcal{N}(v)}^{(l)}\right).$$

After L iterations, a node's representation encodes the structural information within its L-hop neighborhood. The choices of $\text{AGGR}^{(l)}(\cdot)$ and $\text{COMBINE}^{(l)}(.)$ differ by the specific GNN model. An example of $\text{AGGR}^{(l)}(\cdot)$ is the sum of the representations of $\mathcal{N}(v)$. An example of $\text{COMBINE}^{(l)}(\cdot)$ is a concatenation after the feature transformation. In what follows, we describe three important graph neural network models used throughout this thesis.

Graph Convolutional Networks

Kipf and Welling (2017) introduce the Graph Convolutional Network (GCN), a specialized neural architecture with message passing operations that is motivated as a first-order (i.e., linear) approximation to spectral graph convolutions (Hammond et al., 2011). At each layer l, GCN computes the representation of a node $v \in \mathcal{V}$ as follows:

$$\mathbf{g}_v^{(l)} = \sigma\Big(\sum_{u \in \mathcal{N}(v)} \frac{1}{\sqrt{d_v d_u}} \mathbf{W}^{(l)} \mathbf{h}_u^{(l-1)}\Big), \tag{3.8}$$

where $\mathcal{N}(v)$ is a set of nodes with incoming edges to v and v itself, d_v is the degree of v, $\sigma(\cdot)$ is a non-linear activation function, and $\mathbf{W}^{(l)} \in \mathbb{R}^{d^{(l)} \times d^{(l-1)}}$ is a model parameter.

Relational Graph Convolutional Networks

Relational Graph Convolution (RGCN) (Schlichtkrull et al., 2018) is a variant of GCN that extends the architecture to consider different edge types between nodes. In particular, at each layer l, the representation of a node $v \in \mathcal{V}$ is computed as:

$$\mathbf{g}_v^{(l)} = \sigma\Big(\sum_{r \in \mathcal{R}} \sum_{u \in \mathcal{N}_r(v)} \frac{1}{|\mathcal{N}_r(v)|} \mathbf{W}_r^{(l)} \mathbf{h}_u^{(l-1)}\Big), \tag{3.9}$$

where $\mathcal{R}$ denotes the set of relations, $\mathcal{N}_r(v)$ denotes the set of neighbors under the relation $r \in \mathcal{R}$, and $\mathbf{W}_r^{(l)} \in \mathbb{R}^{d^{(l)} \times d^{(l-1)}}$ encodes the edge type between the nodes u and v. In order to avoid the rapid growth in the number of parameters with regard to the number of relations in the graph, regularization based on basis functions and block decompositions can be employed (Schlichtkrull et al., 2018).

Graph Attention Networks

Graph Attention Networks (GAT) (Veličković et al., 2018) employ attentive mechanisms to improve the exploitation of non-trivial graph structure. They encode node

representations by attending over their neighbors, following a self-attention strategy:

$$\mathbf{h}_i^{(l)} = \sigma\Big(\alpha_{ii}\mathbf{W}^{(l)}\mathbf{h}_i^{(l-1)} + \sum_{j\in\mathcal{N}(i)} \alpha_{ij}\mathbf{W}^{(l)}\mathbf{h}_j^{(l-1)}\Big), \tag{3.10}$$

where attention coefficients α_{ij} are computed as:

$$\alpha_{ij} = \text{softmax}\left(\sigma\left(\mathbf{a}^{(l)\top}[\mathbf{W}^{(l)}\mathbf{h}_i^{(l-1)} \,\|\, \mathbf{W}^{(l)}\mathbf{h}_j^{(l-1)}]\right)\right), \tag{3.11}$$

where $\|$ denotes concatenation. $\mathbf{W}^{(l)} \in \mathbb{R}^{d^{(l)}\times d^{(l-1)}}$ and $\mathbf{a}^{(l)} \in \mathbb{R}^{2d^{(l)}}$ are model parameters. The virtue of the attention mechanism is its ability to focus on the most important parts of the node neighborhood. In order to learn attention weights from different perspectives, GAT can employ multi-head attention. Note that GAT shares similarities to the Transformer self-attention mechanism, employing multiple heads and attention coefficients. However, GAT aggregates representations based on the node neighborhood, whereas Transformer self-attention aggregates representations from the whole set of elements.

3.3.2 Graph-to-text Architecture

Specialized encoders based on GNNs are shown to have better performance compared to vanilla encoders in different graph-based NLP tasks (Marcheggiani and Titov, 2017; Bastings et al., 2017; Yao et al., 2019), especially when handling complex graph structures. This indicates that explicitly accounting for the input graph structure can assist graph-to-text generation. However, it is challenging to modify the encoder-decoder architecture to better handle graph-based input data, partly because of the structural differences between the input graph and the output text (Zhao et al., 2020). One straightforward way to explicitly model the input graph structure into graph-to-text models is to replace the sequential encoder used in the original encoder-decoder architecture with a multi-layer GNN. Furthermore, multiple GNN architectures can be considered, and combinations of distinct encoders (e.g., sequential encoder and GNN-based encoder) can be developed. In this thesis, we proposed novel approaches, based on GNN and Transformer architectures, for suitable modeling of graph structures for improved graph-to-text generation.

3.3.3 Our Contributions

As presented at the top of Figure 1.1, our contributions towards proposing improved graph encoding models for graph-to-text generation are three-fold: First, we propose a bipartite graph-to-text model based on different GNNs, that explores complementary top-down and bottom-up traversals of the input AMR graph and demonstrate concrete improvements over sequential and previous graph encoder methods. However, GNNs can also face limitations when encoding relations between distant nodes (Xu et al., 2018a; Alon and Yahav, 2021). Then, to overcome the limitations of GNN encoders based on local neighborhood node aggregation, we propose a unified framework that allows combining different strategies for encoding global and local node contexts and demonstrate the framework's applicability on KG-to-text generation. Finally, we develop a novel mechanism to inject the graph structure

of an input graph into a Transformer encoder, modifying the self-attention mechanism. In what follows, we describe the main contributions and results in each of those parts.

In Chapter 4, we propose a dual graph encoder for AMR-to-text generation that encodes distinct perspectives of the AMR graph structure. The model learns parallel representations of nodes, capturing contrasting views of the graph. We further investigate the use of different node message passing strategies, employing different GNN variants to compute node representations based on incoming and outgoing perspectives. In summary, our contributions in Chapter 4 are the following:

- We present a novel architecture for AMR-to-text generation which explicitly encodes two separate top-down and bottom-up views of the input graph. This dual approach creates different massage passing channels in the graph, allowing the model to better encode distinct interactions between nodes.

- We incorporate in our architecture three graph encoders (Gated Graph Neural Networks, Graph Attention Networks and Graph Isomorphic Networks) that have not been studied so far for AMR-to-text generation.

- We propose an evaluation of the generated outputs employing an entailment model, estimating whether a generated sentence (hypothesis) is semantically entailed by the reference sentence (premise) and vice-versa.

- We investigate the performance of the proposed graph-to-text approach concerning different model configurations and data properties, including model capacity, graph size, graph diameter, node out-degree and sentence length.

In Chapter 5, we present a novel framework for KG-to-text generation that encodes the input graph integrating global and local node contexts, in order to learn richer contextualized node embeddings. Global node encoding allows direct communication between two distant nodes, neglecting graph topology as all nodes are directly connected. In contrast, local node encoding considers the connections between neighbor nodes considering the graph structure, but it can fail to capture long-range associations. A combination of both strategies allows direct communication between distant nodes while preserving the graph connectivity. In summary, our contributions presented in Chapter 5 are the following:

- We present a unified graph-to-text framework based on Graph Attention Networks that incorporates both global and local node aggregations, gathering the benefits from both techniques. As part of this framework, we empirically investigate two main architectures: a cascaded approach that performs global node aggregation before performing local node aggregation and a parallel architecture that performs simultaneously global and local aggregations. To further consider fine-grained integration, we also explore layer-wise integration of the global and local graph encoders.

- We demonstrate that the global module mainly focuses on distant nodes in-

stead of the neighbors and closest nodes, whereas the local encoder focuses on the local neighborhood. This suggests that a combination of global and local interactions between entities in a KG is beneficial for KG-to-text generation, improving the overall performance.

- We establish stronger baselines for KG-to-text generation by exploring complementary model configurations using attention mechanisms. Our framework provides a flexible approach that can be adapted to different graph-to-text generation tasks.

In Chapter 6, we mitigate the effects of the graph linearization in Transformer models, developing a new position mechanism for incorporating the graph structure of an input graph into the architecture. Our approach learns to weigh node-node interactions differently for distinct attention heads, thus virtually discovering differently connected views of the input graph. To summarize, our contributions in Chapter 6 are as follows:

- We propose a novel Transformer-based graph-to-text architecture in which shortest path lengths are interpreted as relative position information within a graph self-attention mechanism.

- We find that our approach learned different attention heads for local and global graph information. Interestingly, direct neighbors are considered particularly important even without injecting any structural bias, such as those introduced by a GNN.

- Experiments indicate that our proposed model is able to differentiate between distant but connected nodes and truly unreachable nodes, demonstrating that the proposed mechanism can automatically learn about the graph connectivity.

3.4 Graph-to-Text Generation with Pretrained Language Models

3.4.1 Pretrained Language Models

Pretrained language models (PLMs) (Devlin et al., 2019; Peters et al., 2018) have been increasingly popular in recent years in NLP. Those models have been proven to be capable of encoding vast amounts of linguistic knowledge from corpora into large-scale parameters and learning universal and contextual language representations using carefully defined pretraining objectives for language modeling. PLMs employ a transfer learning paradigm: they are first pretrained in large-scale corpus and then fine-tuned in different downstream tasks. These approaches have considerably outperformed prior state of the art in various NLP tasks without substantial task-specific architecture modifications (Devlin et al., 2019; Yang et al., 2019; Liu et al., 2020b; Radford et al., 2019).

Following this trend, researchers have developed a variety of methods for address-

ing text generation tasks using PLMs. Recent works (Harkous et al., 2020; Mager et al., 2020) apply transfer learning to data-to-text generation, where a pretrained decoder-only generation model, also known as autoregressive model (Radford et al., 2019), is fine-tuned on the target task using structured data. Encoder-decoder approaches also benefit from pretraining in large-scale data. Recently, much research has been done employing different pretraining objectives for Transformer-based encoder-decoder models (Zhang et al., 2020; Qi et al., 2020; Xue et al., 2021). In what follows, we present two pretrained encoder-decoder models used in this thesis:

- BART (Lewis et al., 2020) is pretrained as a text-to-text denoising autoencoder. The input text is corrupted with a random noise function, and then BART is trained to reconstruct the original text. Specifically, this is accomplished by minimizing a reconstruction loss – the cross-entropy between the decoder output and the original text. Transformations in the input text employed in the pretraining phase include masking random tokens, deleting random tokens or a span of k tokens with a single mask token, changing the ordering of the original sentences, and rotating the document to make it start at a specific token. The training corpus is a combination of books and Wikipedia data. BART is particularly effective when fine-tuned for text generation tasks but also has a strong performance in comprehension tasks, such as question answering.

- T5 (Raffel et al., 2020) follows the original Transformer architecture with a modification regarding the positional embeddings, which are learned at each layer. T5 extends the text-to-text architecture to a wide range of NLP problems. In particular, it converts NLP tasks into a text-to-text format using specified prefixes such as "summarize:" for summarization and "question:" for question answering. The pretraining includes supervised and self-supervised strategies. Supervised training is performed on downstream tasks from the GLUE (Wang et al., 2018) and SuperGLUE (Wang et al., 2019) benchmarks, transforming the datasets into the text-to-text format. Self-supervised training randomly removes 15% of the tokens and replaces them with individual sentinel tokens. If a group of tokens is marked for removal, the entire group is substituted with a single sentinel token. While the encoder's input is the corrupted text, the input of the decoder is the original text, and then the target text is composed of the removed tokens delimited by the corresponding sentinel tokens.

Those pretrained models also have multilingual versions that use basically the same architecture but are pretrained in diverse languages. While mBART (Liu et al., 2020a) is pretrained by denoising texts in 25 languages and was initially developed for supervised and unsupervised machine translation, mT5 (Xue et al., 2021) is trained using general-purpose text-to-text format on 101 languages. Those approaches can be fine-tuned using a specific language or using a combination of languages in both supervised and unsupervised scenarios, without any task-specific or language-specific modifications or initialization schemes. For example, distinct

multilingual setups enable transfer learning to other language pairs that were not present in the pretraining corpus.

Graph-based inputs are different in nature from natural language. As a result, knowledge learned from large-scale pretraining using natural language text intuitively cannot be fully transferred to graph-based data in a fine-tuning phase. Accordingly, we are interested in properly adapting those pretrained models to the task of graph-to-text generation, exploiting their text generation fluency capacities while maintaining graph encoding abilities, which are shown to be beneficial. In the next section, we introduce our contributions for adapting models originally pretrained on text for graph-based data.

3.4.2 Our Contributions

As summarized at the bottom of Figure 1.1, our contributions towards adapting PLMs to graph-based inputs comprehend task-adaptive pretraining strategies, structural adapter architectures, and multi-task approaches for multilingual generation from graph structures. The main contributions and findings in each of those parts are described in the following paragraphs.

In Chapter 7, we investigate the applicability of two encoder-decoder PLMs, namely BART and T5, for graph-to-text generation and analyze the impact of different task-adaptive pretraining strategies for downstream graph-to-text tasks. We present a study across three graph domains: meaning representations, Wikipedia KGs and scientific KGs and show that task-adaptive pretraining strategies are beneficial for adapting those models to structured inputs. To summarize, our contributions in Chapter 7 are as follows:

- We propose two task-adaptive pretraining strategies to adapt PLMs to graph-to-text tasks, exploring language model adaptation and supervised task adaptation pretraining with additional task-specific data.
- Our approaches consistently outperform the state of the art by significant margins on three established graph-to-text benchmarks from different domains, exceeding specialized graph architectures without pretraining (e.g., GNNs).
- We show that pretrained approaches for graph-to-text generation produces texts with significantly better fluency than existing systems and human references in a crowdsourcing study.
- We find that PLMs perform well when trained on a shuffled linearized graph representation (bag of node and edge labels), without any information about connectivity, in different KG-to-text datasets. This is unexpected since prior studies demonstrated that explicitly encoding the graph structure enhances models trained from scratch. This suggests that the PLMs benefit from similar facts seen during pretraining or fine-tuning, such that they perform well even when the input graph is reduced to a simple bag of node and edge labels.

A consequent work (Xie et al., 2022) propose an unifying framework for structural knowledge grounding, where the inputs or outputs are heterogeneous. The framework unifies 21 generation tasks, which includes data-to-text, data-to-data and text-to-data formats. It fine-tunes a T5 model employing multi-task learning using prefix-tuning (Li and Liang, 2021). The model, which is trained jointly in a diverse set of structural generation tasks, consistently improves performance on various datasets containing structured data.

In Chapter 8, we propose a novel structure-aware adapter method that allows to inject the input graph structure into PLMs. Adapters (Houlsby et al., 2019; Rücklé et al., 2021) only train a newly introduced set of weights at every layer, instead of fully fine-tuning the entire PLM, thereby sharing the majority of parameters between tasks. Our method employs a *graph convolution* based on GNNs in order to learn representations built upon the graph connectivity over the PLM encoder. Because the adapter mechanism is added to each encoder layer, deep integration of linguistic knowledge and graph knowledge can be achieved. During fine-tuning, only the structural adapters' parameters are updated, whereas the PLM parameters remain unchanged, in contrast to previous graph-to-text methods based on the graph linearizations that fine-tune all model parameters. To summarize, our contributions in Chapter 8 are the following:

- We propose a novel structure-aware adapter approach that directly injects the input graph structure into pretrained models. The main idea is to add layer-wise modules, which extract information from the pretrained layers and make use of it in a graph-structure encoding.

- We evaluate the model on AMR-to-text generation, establishing new state-of-the-art results on two datasets for this task.

- We analyze and compare our model and previous approaches, shedding light on the capabilities that our architecture acquires for the downstream generation task. We show that our approach is more effective when encoding complex graphs and when trained on a limited amount of data.

- An essential benefit of modeling the graph structure is to be less dependent on linearization techniques because the graph connectivity is invariant to the graph linearization. Our experiments demonstrate that our adapter approach is more robust to different graph linearizations and node reentrancies (nodes with more than one entering edge).

Scientific advancement in text generation has frequently concentrated on enhancing the performance of systems that solely operate in English for a variety of complex reasons. However, billions of people worldwide speak languages other than English, and most text generation models can be adapted to different languages. In Chapter 9, we explore AMR-to-text generation in a multilingual scenario where the goal is generating sentences in different languages given the same AMR as input. Since AMR can be employed as an intermediate meaning representation in diverse NLP tasks (Lim et al., 2020; Bornea et al., 2021), generating text in different lan-

guages from those structures can enable the use of such models in much broader applications. Our contributions in Chapter 9 are three-fold:

- We explore multilingual encoder-decoder pretrained language models for multilingual generation from graph-based data. In particular, we propose multitask training strategies that efficiently combine structured training data from different languages improving multilingual AMR-to-text generation.

- We investigate various techniques for automatically generating AMR annotations to determine which data source produces the best multilingual performance. First, we parse English sentences into silver AMRs from parallel multilingual corpora, resulting in a dataset consisting of grammatically correct sentences with noisy AMR structures. Second, we leverage machine translation and translate English sentences from the gold AMR-to-text corpus to the respective target languages, resulting in a dataset with correct AMR structures but potentially unfaithful or non-grammatical sentences. Third, we experiment with utilizing the AMR-to-text corpus with both gold English AMR and sentences in multi-source scenarios to enhance multilingual training.

- Our experiments empirically validated that both sources of silver data – silver AMR with gold sentences and gold AMR with silver sentences – are complementary, and a combination of them leads to strong multilingual AMR-to-text generation models. Those results highlight the potential of using AMR as a meaning representation to represent information that can be verbalized in multiple languages.

Part II

Publications

Chapter 4

Enhancing AMR-to-Text Generation with Dual Graph Representations

Enhancing AMR-to-Text Generation with Dual Graph Representations

Leonardo F. R. Ribeiro[†], **Claire Gardent**[‡] **and Iryna Gurevych**[†]

[†]Research Training Group AIPHES and UKP Lab, Technische Universität Darmstadt
www.ukp.tu-darmstadt.de

[‡]CNRS/LORIA, Nancy, France
claire.gardent@loria.fr

Abstract

Generating text from graph-based data, such as Abstract Meaning Representation (AMR), is a challenging task due to the inherent difficulty in how to properly encode the structure of a graph with labeled edges. To address this difficulty, we propose a novel graph-to-sequence model that encodes different but complementary perspectives of the structural information contained in the AMR graph. The model learns parallel top-down and bottom-up representations of nodes capturing contrasting views of the graph. We also investigate the use of different node message passing strategies, employing different state-of-the-art graph encoders to compute node representations based on incoming and outgoing perspectives. In our experiments, we demonstrate that the dual graph representation leads to improvements in AMR-to-text generation, achieving state-of-the-art results on two AMR datasets[1].

1 Introduction

Abstract Meaning Representation (AMR; Banarescu et al. (2013)) is a linguistically-grounded semantic formalism that represents the meaning of a sentence as a rooted directed graph, where nodes are concepts and edges are semantic relations. As AMR abstracts away from surface word strings and syntactic structure producing a language neutral representation of meaning, its usage is beneficial in many semantic related NLP tasks, including text summarization (Liao et al., 2018) and machine translation (Song et al., 2019).

The purpose of AMR-to-text generation is to produce a text which verbalises the meaning encoded by an input AMR graph. This is a challenging task as capturing the complex structural information stored in graph-based data is not trivial, as these are non-Euclidean structures, which implies that properties such as global parametrization, vector space structure, or shift-invariance do not hold (Bronstein et al., 2017). Recently, Graph Neural Networks (GNNs) have emerged as a powerful class of methods for learning effective graph latent representations (Xu et al., 2019) and graph-to-sequence models have been applied to the task of AMR-to-text generation (Song et al., 2018; Beck et al., 2018; Damonte and Cohen, 2019; Guo et al., 2019).

In this paper, we propose a novel graph-to-sequence approach to AMR-to-text generation, which is inspired by pre-neural generation algorithms. These approaches explored alternative (top-down, bottom-up and mixed) traversals of the input graph and showed that a hybrid traversal combining both top-down (TD) and bottom-up (BU) information was best as this permits integrating both global constraints top-down from the input and local constraints bottom-up from the semantic heads (Shieber et al., 1990; Narayan and Gardent, 2012).

Similarly, we present an approach where the input graph is represented by two separate structures, each representing a different view of the graph. The nodes of these two structures are encoded using separate graph encoders so that each concept and relation in the input graph is assigned both a TD and a BU representation.

Our approach markedly differs from existing graph-to-sequence models for MR-to-Text generation (Marcheggiani and Perez Beltrachini, 2018; Beck et al., 2018; Damonte and Cohen, 2019) in that these approaches aggregate all the immediate neighborhood information of a node in a single representation. By exploiting parallel and complementary vector representations of the AMR graph, our approach eases the burden on the neural model in encoding nodes (concepts) and edges (relations) in a single vector representation. It also elimi-

[1]Code is available at https://github.com/UKPLab/emnlp2019-dualgraph

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 3183–3194, Hong Kong, China, November 3–7, 2019.

nates the need for additional positional information (Beck et al., 2018) which is required when the same graph is used to encode both TD and BU information, thereby making the edges undirected.

Our main contributions are the following:

- We present a novel architecture for AMR-to-text generation which explicitly encodes two separate TD and BU views of the input graph.
- We show that our approach outperforms recent AMR-to-text generation models on two datasets, including a model that leverages additional syntactic information (Cao and Clark, 2019).
- We compare the performance of three graph encoders, which have not been studied so far for AMR-to-text generation.

2 Related Work

Early works on AMR-to-text generation employ statistical methods (Flanigan et al., 2016b; Pourdamghani et al., 2016; Castro Ferreira et al., 2017) and apply linearization of the graph by means of a depth-first traversal.

Recent neural approaches have exhibited success by linearising the input graph and using a sequence-to-sequence architecture. Konstas et al. (2017) achieve promising results on this task. However, they strongly rely on named entities anonymisation. Anonymisation requires an ad hoc procedure for each new corpus. The matching procedure needs to match a rare input item correctly (e.g., "United States of America") with the corresponding part in the output text (e.g., "USA") which may be challenging and may result in incorrect or incomplete delexicalisations. In contrast, our approach omits anonymisation. Instead, we use a copy mechanism (See et al., 2017), a generic technique which is easy to integrate in the encoder-decoder framework and can be used independently of the particular domain and application. Our approach further differs from Konstas et al. (2017) in that we build a dual TD/BU graph representation and use graph encoders to represent nodes.

Cao and Clark (2019) factor the generation process leveraging syntactic information to improve the performance. However, they linearize both AMR and constituency graphs, which implies that important parts of the graphs cannot well be represented (e.g., coreference).

Several graph-to-sequence models have been proposed. Marcheggiani and Perez Beltrachini (2018) show that explicitly encoding the structure of the graph is beneficial with respect to sequential encoding. They evaluate their model on two tasks, WebNLG (Gardent et al., 2017) and SR11Deep (Belz et al., 2011), but do not apply it to AMR benchmarks. Song et al. (2018) and Beck et al. (2018) apply recurrent neural networks to directly encode AMR graphs. Song et al. (2018) use a graph LSTM as the graph encoder, whereas Beck et al. (2018) develop a model based on GRUs. We go a step further in that direction by developing parallel encodings of graphs which are able to highlight different graph properties.

In a related task, Koncel-Kedziorski et al. (2019) propose an attention-based graph model that generates sentences from knowledge graphs. Schlichtkrull et al. (2018) use Graph Convolutional Networks (GCNs) to tackle the tasks of link prediction and entity classification on knowledge graphs.

Damonte and Cohen (2019) show that off-the-shelf GCNs cannot achieve good performance for AMR-to-text generation. To tackle this issue, Guo et al. (2019) introduce dense connectivity to GNNs in order to integrate both local and global features, achieving good results on the task. Our work is related to Damonte and Cohen (2019), that use stacking of GCN and LSTM layers to improve the model capacity and employ anonymization. However, our model is substantially different: (i) we learn dual representations capturing top-down and bottom-up adjuvant views of the graph, (ii) we employ more effective graph encoders (with different neighborhood aggregations) than GCNs and (iii) we employ copy and coverage mechanisms and do not resort to entity anonymization.

3 Graph-to-Sequence Model

In this section, we describe (i) the representations of the graph adopted as inputs, (ii) the model architecture, including the Dual Graph Encoder and (iii) the GNNs employed as graph encoders.

3.1 Graph Preparation

Let $G = (V, E, R)$ denote a rooted and directed AMR graph with nodes $v_i \in V$ and labeled edges $(v_i, r, v_j) \in E$, where $r \in R$ is a relation type.

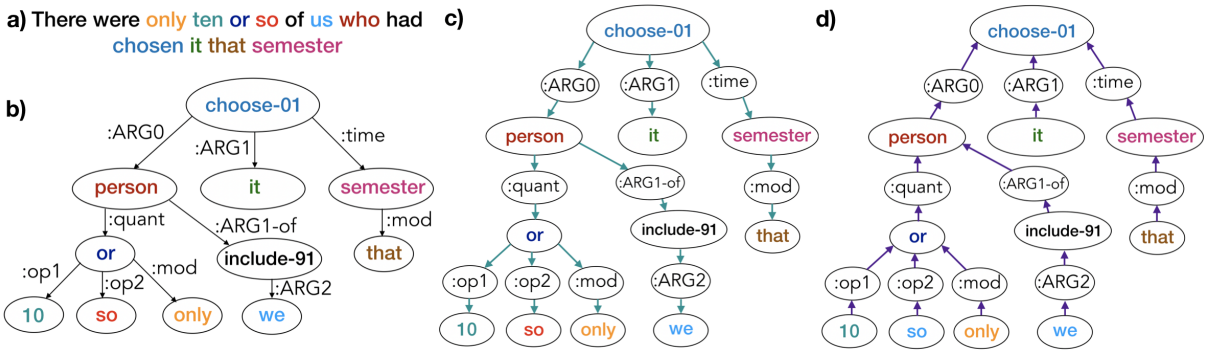


Figure 1: (a) an example sentence, (b) its original AMR graph (G) and different graph perspectives: (c) top-down (G_t) and (d) bottom-up (G_b).

Let $n = |V|$ and $m = |E|$ denote the numbers of nodes and edges, respectively.

We convert each AMR graph into an unlabeled and connected bipartite graph $G_t = (V_t, E_t)$, transforming each labeled edge $(v_i, r, v_j) \in E$ into two unlabeled edges $(v_i, r), (r, v_j) \in E_t$, with $|V_t| = n + m$ and $|E_t| = 2m$. This process, called Levi Transformation (Beck et al., 2018), turns original edges into nodes creating an unlabeled graph. For instance, the edge between `semester` and `that` with label `:mod` in Figure 1(b) is replaced by two edges and one node in 1(c): an edge between `semester`, and the new node `:mod` and another one between `:mod` and `that`. The new graph allows us to directly represent the relationships between nodes using embeddings. This enables us to encode label edge information using distinct message passing schemes employing different GNNs.

G_t captures a TD view of the graph. We also create a BU view of the graph $G_b = (V_t, E_b)$, where each directed edge $e_k = (v_i, v_j) \in E_t$ becomes $e_k = (v_j, v_i) \in E_b$, that is, we reverse the direction of original edges. An example of a sentence, its AMR graph and the two new graphs G_t and G_b is shown in Figure 1.

3.2 Dual Graph Encoder

We represent each node $v_i \in V_t$ with a node embedding $\mathbf{e}_i \in \mathbb{R}^d$, generated from the node label. In order to explicitly encode structural information, our encoder starts with two graph encoders, denoted by GE_t and GE_b, that compute representations for nodes in G_t and G_b, respectively.

Each GE learns node representations based on the specific view of its particular graph, G_t or G_b. Since G_t and G_b capture distinct perspectives of the graph structure, the information flow is propagated throughout TD and BU directions, respectively. In particular, for each node v_i, the GE receives the node embeddings of v_i and its neighbors, and computes its node representation:

$$\mathbf{h}_i^t = GE_t(\{\mathbf{e}_i, \mathbf{e}_j : j \in \mathcal{N}_t(i)\}),$$
$$\mathbf{h}_i^b = GE_b(\{\mathbf{e}_i, \mathbf{e}_j : j \in \mathcal{N}_b(i)\}),$$

where $\mathcal{N}_t(i)$ and $\mathcal{N}_b(i)$ are the immediate incoming neighborhoods of v_i in G_t and G_b, respectively.

Each node v_i is represented by two different hidden states, $\mathbf{h}_i^t$ and $\mathbf{h}_i^b$. Note that we learn two representations per relation and node of the original AMR graph. The hidden states $\mathbf{h}_i^t$ and $\mathbf{h}_i^b$, and embedding $\mathbf{e}_i$ contain different information regarding v_i. We concatenate them building a final node representation:

$$\mathbf{r}_i = \left[\mathbf{h}_i^t \,\|\, \mathbf{h}_i^b \,\|\, \mathbf{e}_i\right].$$

This approach is similar to bidirectional RNNs (Schuster and Paliwal, 1997). Bidirectional RNNs benefit from left-to-right and right-to-left propagation. They learn the hidden representations separately and concatenate them at the end. We perform a similar encoding: first we learn TD and BU representations independently, and lastly, we concatenate them.

The final representation $\mathbf{r}_i$ is employed in a sequence input of a bidirectional LSTM. For each AMR graph, we generate a node sequence by depth-first traversal order. In particular, given a representation sequence from $\mathbf{r}_1$ to $\mathbf{r}_n$, the hidden forward and backward states of $\mathbf{r}_i$ are defined as:

$$\overrightarrow{\mathbf{h}}_i = LSTM_f(\mathbf{r}_i, \overrightarrow{\mathbf{h}}_{i-1}),$$
$$\overleftarrow{\mathbf{h}}_i = LSTM_b(\mathbf{r}_i, \overleftarrow{\mathbf{h}}_{i-1}),$$

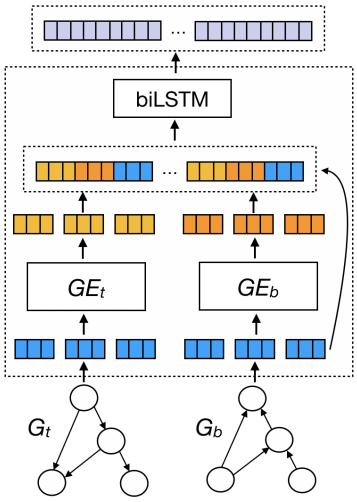


Figure 2: Dual Graph Encoder. The encoder receives the two graph views and generates structural node representations that are used by the decoder. Representations in blue, yellow and orange are $\mathbf{e}_i$, $\mathbf{h}_i^t$ and $\mathbf{h}_i^b$, respectively.

where $LSTM_f$ is a forward LSTM and $LSTM_b$ is a backward LSTM. Note that, for the backward LSTM, we feed the reversed input as the order from $\mathbf{r}_n$ to $\mathbf{r}_1$. Lastly, we obtain the final hidden state by concatenating them as:

$$\mathbf{h}_i = [\overrightarrow{\mathbf{h}}_i \,\|\, \overleftarrow{\mathbf{h}}_i].$$

The resulting hidden state $\mathbf{h}_i$ encodes the information of both preceding and following nodes.

Stacking layers was demonstrated to be effective in graph-to-sequence approaches (Marcheggiani and Perez Beltrachini, 2018; Koncel-Kedziorski et al., 2019; Damonte and Cohen, 2019) and allows us to test for their contributions to the system performance more easily. We employ different GNNs for both graph encoders (Section 3.3). Figure 2 shows the proposed encoder architecture.

3.3 Graph Neural Networks

The GEs incorporate, in each node representation, structural information based on both views of the graph. We explore distinct strategies for neighborhood aggregation, adopting three GNNs: Gated Graph Neural Networks (GGNN, Li et al. (2016)), Graph Attention Networks (GAT, Veličković et al. (2018)) and Graph Isomorphic Networks (GIN, Xu et al. (2019)). Each GNN employs a specific message passing scheme which allows capturing different nuances of structural information.

Gated Graph Neural Networks GGNNs employ gated recurrent units to encode node representations, reducing the recurrence to a fixed number of steps. In particular, the l-th layer of a GGNN is calculated as:

$$\mathbf{h}_i^{(l)} = GRU\Big(\mathbf{h}_i^{(l-1)}, \sum_{j \in \mathcal{N}(i)} \mathbf{W_1} \mathbf{h}_j^{(l-1)}\Big),$$

where $\mathcal{N}(i)$ is the immediate neighborhood of v_i, $\mathbf{W_1}$ is a parameter and GRU is a gated recurrent unit (Cho et al., 2014). Different from other GNNs, GGNNs use back-propagation through time (BPTT) to learn the parameters. GGNNs also do not require to constrain parameters to ensure convergence.

Graph Attention Networks GATs apply attentive mechanisms to improve the exploitation of non-trivial graph structure. They encode node representations by attending over their neighbors, following a self-attention strategy:

$$\mathbf{h}_i^{(l)} = \alpha_{i,i} \mathbf{W_2} \mathbf{h}_i^{(l-1)} + \sum_{j \in \mathcal{N}(i)} \alpha_{i,j} \mathbf{W_2} \mathbf{h}_j^{(l-1)},$$

where attention coefficients $\alpha_{i,j}$ are computed as:

$$\alpha_{i,j} = \text{softmax}\left(\sigma\left(\mathbf{a}^\top [\mathbf{W_2}\mathbf{h}_i^{(l-1)} \,\|\, \mathbf{W_2}\mathbf{h}_j^{(l-1)}]\right)\right),$$

where σ is the activation function and $\|$ denotes concatenation. $\mathbf{W_2}$ and $\mathbf{a}$ are model parameters. The virtue of the attention mechanism is its ability to focus on the most important parts of the node neighborhood. In order to learn attention weights in different perspectives, GATs can employ multi-head attentions.

Graph Isomorphic Networks GIN is a GNN as powerful as the Weisfeiler-Lehman (WL) graph isomorphism test (Weisfeiler and Lehman, 1968) in representing isomorphic and non-isomorphic graphs with discrete attributes. Its l-th layer is defined as:

$$\mathbf{h}_i^{(l)} = h_{\mathbf{W}}\Big(\mathbf{h}_i^{(l-1)} + \sum_{j \in \mathcal{N}(i)} \mathbf{h}_j^{(l-1)}\Big),$$

where $h_{\mathbf{W}}$ is a multi-layer perceptron (MLP). In contrast to other GNNs, which combine node feature with its aggregated neighborhood feature, GINs do not apply the combination step and simply aggregate the node along with its neighbors.

Each of these GNNs applies different approaches to learn structural features from graph data and has achieved impressive results on many graph-based tasks (Li et al., 2016; Veličković et al., 2018; Xu et al., 2019).

	LDC2015E86			LDC2017T10		
training, dev and test instances	16,833	1,368	1,371	36,521	1,368	1,371
min, average and max graph diameter	0	6.9	20	0	6.7	20
min, average and max node degree	0	2.1	18	0	2.1	20
min, average and max number of nodes	1	17.7	151	1	16.8	151
min, average and max number of edges	0	18.6	172	0	17.7	172
number of DAG and non-DAG graphs	18,679	893		37,284	1,976	
min, average and max length sentences	1	21.3	225	1	20.4	225

Table 1: Data statistics of LDC2015E86 and LDC2017T10 datasets. The values are calculated for all splits (train, development and test sets). DAG stands for directed acyclic graph.

3.4 Decoder

An attention-based unidirectional LSTM decoder is used to generate sentences, attending to the hidden representations of edges and nodes. In each step t, the decoder receives the word embedding of the previous word (during training, this is the previous word of the reference sentence; at test time it is the previously generated word), and has the decoder state $\mathbf{s}_t$. The attention distribution $\mathbf{a}^t$ is calculated as in See et al. (2017):

$$\mathbf{e}_i^t = \mathbf{v} \cdot \tanh(\mathbf{W}_h \mathbf{h}_i + \mathbf{W}_s \mathbf{s}_t + \mathbf{w}_c \mathbf{s}_c + \mathbf{b}),$$
$$\mathbf{a}^t = \text{softmax}(\mathbf{e}^t),$$

where $\mathbf{s}_c$ is the coverage vector and $\mathbf{v}$, $\mathbf{W}_h$, $\mathbf{W}_s$, $\mathbf{w}_c$ and $\mathbf{b}$ are learnable parameters. The coverage vector is the accumulation of all attention distributions so far.

Copy and Coverage Mechanisms Previous works (Damonte and Cohen, 2019; Cao and Clark, 2019) use anonymization to handle names and rare words, alleviating the data sparsity. In contrast, we employ copy and coverage mechanisms to address out-of-vocabulary issues for rare target words and to avoid repetition (See et al., 2017).

The model is trained to optimize the negative log-likelihood:

$$\mathcal{L} = -\sum_{t=1}^{|Y|} \log\, p(y_t | y_{1:t-1}, X; \theta),$$

where $Y = y_1, \ldots, y_{|Y|}$ is the sentence, X is the AMR graph and θ represents the model parameters.

4 Data

We use two AMR corpora, LDC2015E86 and LDC2017T10[2]. In these datasets, each instance contains an AMR graph and a sentence. Table 1 shows the statistics for both datasets. Figure 3 shows the distribution of the AMR graph diameters and node degrees for both datasets. The AMR graph structures are similar for most examples. Note that 90% of AMR graphs in both datasets have the diameter less than or equal to 11 and 90% of nodes have the degree of 4 or less. Very structurally similar graphs pose difficulty for the graph encoder by making it harder to learn the differences between their similar structures. Therefore, the word embeddings used as additional input play an important role in helping the model to deal with language information. That is one of the reasons why we concatenate this information in the node representation $\mathbf{r}_i$.

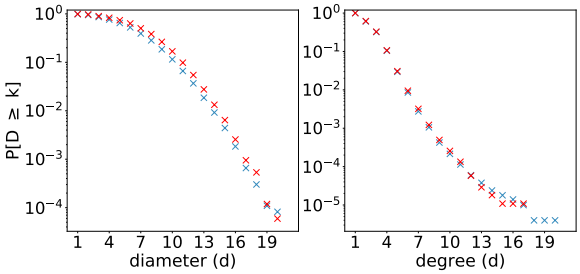


Figure 3: Distribution of the AMR graph diameter (left) and node degree (right) in the training set for LDC2015E86 (red) and LDC2017T10 (blue) datasets.

5 Experiments and Discussion

Implementation Details We extract vocabularies (size of 20,000) from the training sets and initialize the node embeddings from GloVe word embeddings (Pennington et al., 2014) on Common Crawl. Hyperparameters are tuned on the development set of the LDC2015E86 dataset. For GIN, GAT, and GGNN graph encoders, we set the number of layers to 2, 5 and 5, respectively. To regu-

[2]The datasets can be found at https://amr.isi.edu/download.html

Model	BLEU	METEOR
LDC2015E86		
Konstas et al. (2017)	22.00	-
Song et al. (2018)	23.28	30.10
Cao et al. (2019)	23.50	-
Damonte et al.(2019)	24.40	23.60
Guo et al. (2019)	**25.70**	-
S2S	22.55 ± 0.17	29.90 ± 0.31
G2S-GIN	22.93 ± 0.20	29.72 ± 0.09
G2S-GAT	23.42 ± 0.16	29.87 ± 0.14
G2S-GGNN	24.32 ± 0.16	**30.53** ± 0.30
LDC2017T10		
Back et al. (2018)	23.30	-
Song et al. (2018)	24.86	31.56
Damonte et al.(2019)	24.54	24.07
Cao et al. (2019)	26.80	-
Guo et al. (2019)	27.60	-
S2S	22.73 ± 0.18	30.15 ± 0.14
G2S-GIN	26.90 ± 0.19	32.62 ± 0.04
G2S-GAT	26.72 ± 0.20	32.52 ± 0.02
G2S-GGNN	**27.87** ± 0.15	**33.21** ± 0.15

Table 2: BLEU and METEOR scores on the test set of LDC2015E86 and LDC2017T10 datasets.

larize the model, during training we apply dropout (Srivastava et al., 2014) to the graph layers with a rate of 0.3. The graph encoder hidden vector sizes are set to 300 and hidden vector sizes for LSTMs are set to 900.

The models are trained for 30 epochs with early stopping based on the development BLEU score. For our models and the baseline, we used a two-layer LSTM decoder. We use Adam optimization (Kingma and Ba, 2015) as the optimizer with an initial learning rate of 0.001 and 20 as the batch size. Beam search with the beam size of 5 is used for decoding.

Results We call the models G2S-GIN (isomorphic encoder), G2S-GAT (graph-attention encoder), and G2S-GGNN (gated-graph encoder), according to the graph encoder utilized. As a baseline (S2S), we train an attention-based encoder-decoder model with copy and coverage mechanisms, and use a linearized version of the graph generated by depth-first traversal order as input. We compare our models against several state-of-the-art results reported on the two datasets (Konstas et al., 2017; Song et al., 2018; Beck et al., 2018; Damonte and Cohen, 2019; Cao and Clark, 2019; Guo et al., 2019).

Model	External	BLEU
Konstas et al. (2017)	200K	27.40
Song et al. (2018)	200K	28.20
Guo et al. (2019)	200K	31.60
G2S-GGNN	200K	**32.23**

Table 3: Results on LDC2015E86 test set when models are trained with additional Gigaword data.

We use both BLEU (Papineni et al., 2002) and METEOR (Denkowski and Lavie, 2014) as evaluation metrics[3]. In order to mitigate the effects of random seeds, we report the averages for 4 training runs of each model along with their standard deviation. Table 2 shows the comparison between the proposed models, the baseline and other neural models on the test set of the two datasets.

For both datasets, our approach substantially outperforms the baselines. In LDC2015E86, G2S-GGNN achieves a BLEU score of 24.32, 4.46% higher than Song et al. (2018), who also use the copy mechanism. This indicates that our architecture can learn to generate better signals for text generation. On the same dataset, we have competitive results to Damonte and Cohen (2019). However, we do not rely on preprocessing anonymisation not to lose semantic signals. In LDC2017T10, G2S-GGNN achieves a BLEU score of 27.87, which is 3.33 points higher than Damonte and Cohen (2019), a state-of-the-art model that does not employ external information. We also have competitive results to Guo et al. (2019), a very recent state-of-the-art model.

We also outperform Cao and Clark (2019) improving BLEU scores by 3.48% and 4.00%, in LDC2015E86 and LDC2017T10, respectively. In contrast to their work, we do not rely on (i) leveraging supplementary syntactic information and (ii) we do not require an anonymization preprocessing step. G2S-GIN and G2S-GAT have comparable performance on both datasets. Interestingly, G2S-GGNN has better performance among our models. This suggests that graph encoders based on gating mechanisms are very effective in text generation models. We hypothesize that the gating mechanism can better capture long-distance dependencies between nodes far apart in the graph.

[3]For BLEU, we use the multi-BLEU script from the MOSES decoder suite (Koehn et al., 2007). For METEOR, we use the original meteor-1.5.jar script (https://github.com/cmu-mtlab/meteor).

Model	BLEU	METEOR	Size
biLSTM	22.50	30.42	57.6M
GE_t + biLSTM	26.33	32.62	59.6M
GE_b + biLSTM	26.12	32.49	59.6M
GE_t + GE_b + biLSTM	27.37	33.30	61.7M

Table 4: Results of the ablation study on the LDC2017T10 development set.

Additional Training Data Following previous works (Konstas et al., 2017; Song et al., 2018; Guo et al., 2019), we also evaluate our models employing additional data from English Gigaword corpus (Napoles et al., 2012). We sample 200K Gigaword sentences and use JAMR[4] (Flanigan et al., 2016a) to parse them. We follow the method of Konstas et al. (2017), which is fine-tuning the model on the LDC2015E86 training set after every epoch of pretraining on the Gigaword data. G2S-GGNN outperforms others with the same amount of Gigaword sentences (200K), achieving a 32.23 BLEU score, as shown in Table 3. The results demonstrate that pretraining on automatically generated AMR graphs enhances the performance of our model.

Ablation Study In Table 4, we report the results of an ablation study on the impact of each component of our model on the development set of LDC2017T10 dataset by removing the graph encoders. We also report the number of parameters (including embeddings) used in each model. The first thing we notice is the huge increase in metric scores (17% in BLEU) when applying the graph encoder layer, as the neural model receives signals regarding the graph structure of the input. The dual representation helps the model with a different view of the graph, increasing BLEU and METEOR scores by 1.04 and 0.68 points, respectively. The complete model has slightly more parameters than the model without graph encoders (57.6M vs 61.7M).

Impact of Graph Size, Arity and Sentence Length The good overall performance on the datasets shows the superiority of using graph encoders and dual representations over the sequential encoder. However, we are also interested in estimating the performance of the models concerning different data properties. In order to evaluate how the models handle graph and sentence features, we perform an inspection based on different sizes of graph diameter, sentence length, and max node out-degree. Table 5 shows METEOR[5] scores for the LDC2017T10 dataset.

Model	Graph Diameter					
	0-7	Δ	7-13	Δ	14-20	Δ
S2S	33.2		29.7		28.8	
G2S-GIN	35.2	+6.0%	31.8	+7.4%	31.5	+9.2%
G2S-GAT	35.1	+5.9%	32.0	+7.8%	31.5	+9.51%
G2S-GGNN	36.2	+9.0%	33.0	+11.4%	30.7	+6.7%
	Sentence Length					
	0-20	Δ	20-50	Δ	50-240	Δ
S2S	34.9		29.9		25.1	
G2S-GIN	36.7	+5.2%	32.2	+7.8%	26.5	+5.8%
G2S-GAT	36.9	+5.7%	32.3	+7.9%	26.6	+6.1%
G2S-GGNN	37.9	+8.5%	33.3	+11.2%	26.9	+6.8%
	Max Node Out-degree					
	0-3	Δ	4-8	Δ	9-18	Δ
S2S	31.7		30.0		23.9	
G2S-GIN	33.9	+6.9%	32.1	+6.9%	25.4	+6.2%
G2S-GAT	34.3	+8.0%	32.0	+6.7%	22.5	-6.0%
G2S-GGNN	35.0	+10.3%	33.1	+10.4%	22.2	-7.3%

Table 5: METEOR scores and differences to the S2S, in the LDC2017T10 test set, with respect to the graph diameter, sentence length and max node out-degree.

The performances of all models decrease as the diameters of the graphs increase. G2S-GGNN has a 17.9% higher METEOR score in graphs with a diameter of at most 7 compared to graphs with diameters higher than 13. This is expected as encoding a bigger graph (containing more information) is harder than encoding smaller graphs. Moreover, 71% of the graphs in the training set have a diameter less than or equal to 7 and only 2% have a diameter bigger than 13 (see Figure 3). Since the models have fewer examples of bigger graphs to learn from, this also leads to worse performance when handling graphs with higher diameters. We also investigate the performance with respect to the sentence length. The models have better results when handling sentences with 20 or fewer tokens. Longer sentences pose additional challenges to the models.

G2S-GIN has a better performance in handling graphs with node out-degrees higher than 9. This indicates that GINs can be employed in tasks where the distribution of node degrees has a long

[4] https://github.com/jflanigan/jamr

[5] METEOR score is used as it is a sentence-level metric.

	REF ⇒ GEN		
Model	**ENT**	**CON**	**NEU**
S2S	38.45	11.17	50.38
G2S-GIN	49.78	9.80	40.42
G2S-GAT	49.48	8.09	42.43
G2S-GGNN	51.32	8.82	39.86
	GEN ⇒ REF		
Model	**ENT**	**CON**	**NEU**
S2S	73.79	12.75	13.46
G2S-GIN	76.27	10.65	13.08
G2S-GAT	77.54	8.54	13.92
G2S-GGNN	77.64	9.64	12.72

Table 6: Entailment (ENT), contradiction (CON) and neutral (NEU) average percentages for the LDC2017T10 test set. **(Top)** The premise and the hypothesis are the generated (GEN) and reference (REF) sentences, respectively. **(Bottom)** The hypothesis and the premise are the generated (GEN) and reference (REF) sentences, respectively.

tail. Surprisingly, S2S has a better performance than G2S-GGNN and G2S-GAT when handling graphs that contain high degree nodes.

Semantic Equivalence We perform an entailment experiment using BERT (Devlin et al., 2019) fine-tuned on the MultiNLI dataset (Williams et al., 2018) as a NLI model. We are interested in exploring whether a generated sentence (hypothesis) is semantically *entailed* by the reference sentence (premise). In a related text generation task, Falke et al. (2019) employ NLI models to rerank alternative predicted abstractive summaries.

Nevertheless, uniquely verifying whether the reference (REF) entails the generated sentence (GEN) or vice-versa (GEN entails REF) is not sufficient. For example, suppose that *"Today Jon walks"* is the REF and *"Jon walks"* is the GEN. Even though REF entails GEN, GEN does not entail REF, that is, GEN is too general (missing information). Furthermore, suppose that *"Jon walks"* is the REF and *"Today Jon walks"* is the GEN, GEN entails REF but REF does not entail GEN, that is, GEN is too specific (added information). Therefore, in addition to verify whether the reference entails the generated sentence, we also verify whether the generated sentence entails the reference.

Table 6 shows the average probabilities for entailment, contradiction and neutral classes on the LDC2017T10 test set. All G2S models have higher entailment compared to S2S. G2S-GGNN has 33.5% and 5.2% better entailment performances than S2S, when REF entails GEN and GEN entails REF, respectively. G2S models also generate sentences that contradict the reference sentences less. This suggests that our models are capable of capturing better semantic information from the graph generating outputs semantically related to the reference sentences.

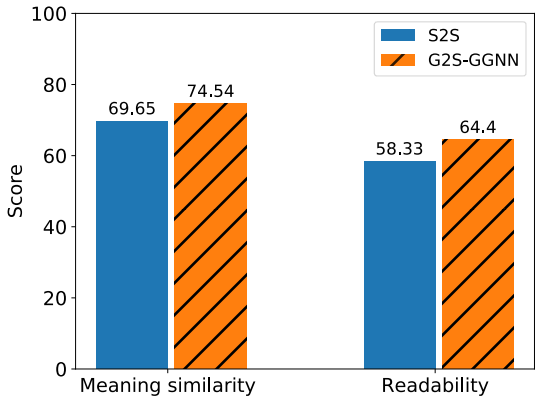


Figure 4: Human evaluation of the sentences generated by S2S and G2S-GGNN models. Results are statistically significant with $p < 0.05$, using Wilcoxon rank-sum test.

Human Evaluation To further assess the quality of the generated sentences, we conduct a human evaluation. We employ the *Direct Assessment* (DA) method (Graham et al., 2017) via Amazon Mechanical Turk. Using the DA method inspired by Mille et al. (2018), we assess two quality criteria: (i) *meaning similarity*: how close in meaning the generated text is to the gold sentence; and (ii) *readability*: how well the generated sentence reads (Is it good fluent English?).

We randomly select 100 sentences generated by S2S and G2S-GGNN and randomly assign them to HITs (following Mechanical Turk terminology). Human workers rate the sentences according to meaning similarity and readability on a 0-100 rating scale. The tasks are executed separately and workers were first given brief instructions. For each sentence, we collect scores from 5 workers and average them. Models are ranked according to the mean of sentence-level scores. We apply a quality control step filtering workers who do not score some faked and known sentences properly.

Figure 4 shows the results. In both metrics, G2S-GGNN has better human scores for meaning similarity and readability, suggesting a higher

(a / agree :ARG0 (a2 / and :op1 (c / country :wiki China :name (n / name :op1 China)) :op2 (c2 / country :wiki Kyrgyzstan :name (n2 / name :op1 Kyrgyzstan))) :ARG1 (t / threaten-01 :ARG0 (a3 / and :op1 (t2 / terrorism) :op2 (s / separatism) :op3 (e / extremism)) :ARG2 (a4 / and :op1 (s3 / security :mod (r / region)) :op2 (s4 / stability :mod r)) :time (s2 / still) :ARG1-of (m / major-02)) :medium (c3 / communique :mod (j / joint)))	
GOLD	China and Kyrgyzstan agreed in a joint communique that terrorism, separatism and extremism still pose major threats to regional security and stability.
S2S	In the joint communique, China and Kyrgyzstan still agreed to threaten terrorism, separatism, extremism and regional stability.
Song et. al (2018)	In a joint communique, China and Kyrgyzstan have agreed to still be a major threat to regional security, and regional stability.
G2S-GGNN	At a joint communique, China and Kyrgyzstan agreed that terrorism, separatism and extremism are still a major threat to region security and stability.

Table 7: An example of an AMR graph and generated sentences. GOLD refers to the reference sentence.

Model	ADDED	MISS
S2S	47.34	37.14
G2S-GIN	48.67	33.64
G2S-GAT	48.24	33.73
G2S-GGNN	48.66	34.06
GOLD	50.77	28.35

Table 8: Fraction of elements in the output that are not present in the input (ADDED) and the fraction of elements in the input graph that are missing in the generated sentence (MISS), for the test set of LDC2017T10. The token lemmas are used in the comparison. GOLD refers to the reference sentences.

quality of the generated sentences regarding S2S. The Pearson correlations between meaning similarity and readability scores, and METEOR[6] scores are 0.50 and 0.22, respectively.

Semantic Adequacy We also evaluate the semantic adequacy of our model (how well does the generated output match the input?) by comparing the number of added and missing tokens that occur in the generated versus reference sentences (GOLD). An added token is one that appears in the generated sentence but not in the input graph. Conversely, a missing token is one that occurs in the input but not in the output. In GOLD, added tokens are mostly function words while missing tokens are typically input concepts that differ from the output lemma. For instance, in Figure 1, *there* and *of* are added tokens while *person* is a missing token. As shown in Table 8, G2S approaches outperform the S2S baseline. G2S-GIN is closest to GOLD with respect to both metrics suggesting that this model is better able to generate novel words to construct the sentence and captures a larger range of concepts from the input AMR graph, covering more information.

Manual Inspection Table 7 shows sentences generated by S2S, Song et al. (2018), G2S-GAT, and the reference sentence. The example shows that our approach correctly verbalises the subject of the embedded clause *"China and ... agreed that terrorism, separatism and extremism$_{SUBJ}$ ... pose major threats to ..."*, while S2S and Song et al. (2018) are fooled by the fact that *agree* frequently takes an infinitival argument which shares its subject (*"China ...$_{SUBJ}$ agreed to threaten / have agreed to be a major threat"*). While this is a single example, it suggests that dual encoding enhances the model ability to take into account the dependencies and the graph structure information, rather than the frequency of n-grams.

6 Conclusion

We have studied the problem of generating text from AMR graphs. We introduced a novel architecture that explicitly encodes two parallel and adjuvant representations of the graph (top-down and bottom-up). We showed that our approach outperforms state-of-the-art results in AMR-to-text generation. We provided an extensive evaluation of our models and demonstrated that they are able to achieve the best performance. In the future, we will consider integrating deep generative graph models to express probabilistic dependencies among AMR nodes and edges.

Acknowledgments

This work has been supported by the German Research Foundation as part of the Research Training Group Adaptive Preparation of Information from Heterogeneous Sources (AIPHES) under grant No. GRK 1994/1.

[6]METEOR score is used as it is a sentence-level metric.

References

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

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

Anja Belz, Mike White, Dominic Espinosa, Eric Kow, Deirdre Hogan, and Amanda Stent. 2011. The first surface realisation shared task: Overview and evaluation results. In *Proceedings of the Generation Challenges Session at the 13th European Workshop on Natural Language Generation*, pages 217–226, Nancy, France. Association for Computational Linguistics.

Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42.

Kris Cao and Stephen Clark. 2019. Factorising AMR generation through syntax. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2157–2163, Minneapolis, Minnesota. Association for Computational Linguistics.

Thiago Castro Ferreira, Iacer Calixto, Sander Wubben, and Emiel Krahmer. 2017. Linguistic realisation as machine translation: Comparing different MT models for AMR-to-text generation. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 1–10, Santiago de Compostela, Spain. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Marco Damonte and Shay B. Cohen. 2019. Structural neural encoders for AMR-to-text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3649–3658, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Tobias Falke, Leonardo F. R. Ribeiro, Prasetya Ajie Utama, Ido Dagan, and Iryna Gurevych. 2019. Ranking generated summaries by correctness: An interesting but challenging application for natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2214–2220, Florence, Italy. Association for Computational Linguistics.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016a. CMU at SemEval-2016 task 8: Graph-based AMR parsing with infinite ramp loss. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1202–1206, San Diego, California. Association for Computational Linguistics.

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

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

Yvette Graham, Timohy Baldwin, Alistair Moffat, and Justin Zobel. 2017. Can machine translation systems be evaluated by the crowd alone. *Natural Language Engineering*, 23(1):3–30.

Zhijiang Guo, Yan Zhang, Zhiyang Teng, and Wei Lu. 2019. Densely connected graph convolutional networks for graph-to-sequence learning. *Transactions*

of the Association for Computational Linguistics, 7:297–312.

Diederick P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diego, CA, USA.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text Generation from Knowledge Graphs with Graph Transformers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2284–2293, Minneapolis, Minnesota. Association for Computational Linguistics.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural amr: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157, Vancouver, Canada. Association for Computational Linguistics.

Yujia Li, Richard Zemel, Marc Brockschmidt, and Daniel Tarlow. 2016. Gated graph sequence neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico.

Kexin Liao, Logan Lebanoff, and Fei Liu. 2018. Abstract meaning representation for multi-document summarization. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1178–1190, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

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

Simon Mille, Anja Belz, Bernd Bohnet, Yvette Graham, Emily Pitler, and Leo Wanner. 2018. The first multilingual surface realisation shared task (SR'18): Overview and evaluation results. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 1–12, Melbourne, Australia. Association for Computational Linguistics.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, AKBC-WEKEX '12, pages 95–100, Stroudsburg, PA, USA. Association for Computational Linguistics.

Shashi Narayan and Claire Gardent. 2012. Structure-driven lexicalist generation. In *Proceedings of COLING 2012*, pages 2027–2042, Mumbai, India. The COLING 2012 Organizing Committee.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Nima Pourdamghani, Kevin Knight, and Ulf Hermjakob. 2016. Generating English from abstract meaning representations. In *Proceedings of the 9th International Natural Language Generation conference*, pages 21–25, Edinburgh, UK. Association for Computational Linguistics.

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

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

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

Stuart M. Shieber, Gertjan van Noord, Fernando C. N. Pereira, and Robert C. Moore. 1990. Semantic-head-driven generation. *Computational Linguistics*, 16(1):30–42.

Linfeng Song, Daniel Gildea, Yue Zhang, Zhiguo Wang, and Jinsong Su. 2019. Semantic neural machine translation using AMR. *Transactions of the Association for Computational Linguistics*, 7:19–31.

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

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

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

Boris Weisfeiler and A.A. Lehman. 1968. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsia*, pages 12–16.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks? In *International Conference on Learning Representations*, New Orleans, LA, USA.

A Generated Sentences

Table 9 shows three examples with sentences generated by S2S, Song et al. (2018), G2S-GGNN, and the reference sentence (GOLD).

B Human Evaluation Setup

- For each quality evaluation task (meaning similarity and readability), we independently sampled 100 generated sentences for each model.
- We created separate HITs for meaning similarity and readability evaluations. Each HIT contains 10 sentences.
- We paid $0.15 per HIT, employing five workers on each. For qualification, workers were required to have over 1000 approved HITs.
- We applied a quality control step. We removed workers who do not achieve a minimum threshold in sentences with known scores.

GOLD	I don't want to be miserable anymore and the longer he is around the more miserable I will be.
S2S	If he was in longer longer, I don't want to miserable and more miserable.
Song et. al (2018)	I don't want to be miserable anymore, and when he is around longer, I'm a miserable miserable.
G2S-GGNN	I don't want to be miserable anymore, and would be more miserable if he was around longer.

GOLD	Colombia is the source of much of the cocaine and heroin sold in the United States.
S2S	Colombia is a source of cocaine, much of cocaine and heroin sales in the United States.
Song et. al (2018)	Colombia is a source of much of much of cocaine and heroin in the United States.
G2S-GGNN	Colombia is a source of much cocaine and heroin and heroin sold in the United States.

GOLD	Discussions between Lula da Silva and Thabo Mbeki would also address new threats to international security such as terrorism, drugs, illegal weapons trafficking and aids.
S2S	Thabo da Silva has also addressed Thabo Mbeki to discuss new threats such as terrorism, drugs, illegal weapons trafficking and aids in international security.
Song et. al (2018)	Lula da Silva's discussion with Thabo also addressed a new threat against Thabo Mbeki and aids, drugs, illegal weapons and illegal weapons of weapon.
G2S-GGNN	Lula da Silva's discussion with Thabo da Silva also addressed new threat such as terrorism, drugs, illegal weapons trafficking and aids.

Table 9: Examples of generated sentences. GOLD refers to the reference sentence.

Chapter 5

Modeling Global and Local Node Contexts for Text Generation from Knowledge Graphs

Modeling Global and Local Node Contexts for Text Generation from Knowledge Graphs

Leonardo F. R. Ribeiro[†], **Yue Zhang**[‡], **Claire Gardent**[§] **and Iryna Gurevych**[†]

[†]Research Training Group AIPHES and UKP Lab, Technische Universität Darmstadt
[‡]School of Engineering, Westlake University, [§]CNRS/LORIA, Nancy, France
ribeiro@aiphes.tu-darmstadt.de, yue.zhang@wias.org.cn
claire.gardent@loria.fr, gurevych@ukp.informatik.tu-darmstadt.de

Abstract

Recent graph-to-text models generate text from graph-based data using either global or local aggregation to learn node representations. *Global node encoding* allows explicit communication between two distant nodes, thereby neglecting graph topology as all nodes are directly connected. In contrast, *local node encoding* considers the relations between neighbor nodes capturing the graph structure, but it can fail to capture long-range relations. In this work, we gather both encoding strategies, proposing novel neural models that encode an input graph combining both global and local node contexts, in order to learn better contextualized node embeddings. In our experiments, we demonstrate that our approaches lead to significant improvements on two graph-to-text datasets achieving BLEU scores of 18.01 on the AGENDA dataset, and 63.69 on the WebNLG dataset for seen categories, outperforming state-of-the-art models by 3.7 and 3.1 points, respectively.[1]

1 Introduction

Graph-to-text generation refers to the task of generating natural language text from input graph structures, which can be semantic representations (Konstas et al., 2017) or knowledge graphs (KGs) (Gardent et al., 2017; Koncel-Kedziorski et al., 2019). Whereas most recent work (Song et al., 2018; Ribeiro et al., 2019; Guo et al., 2019) focuses on generating sentences, a more challenging and interesting scenario emerges when the goal is to generate multisentence texts. In this context, in addition to sentence generation, document planning needs to be handled: The input needs to be mapped into several sentences; sentences need to be ordered and connected using appropriate discourse markers; and inter-sentential anaphora and ellipsis may need to be generated to avoid repetition. In this paper, we focus on generating texts rather than sentences where the output are short texts (Gardent et al., 2017) or paragraphs (Koncel-Kedziorski et al., 2019).

A key issue in neural graph-to-text generation is how to encode the input graphs. The basic idea is to incrementally compute node representations by aggregating structural context information. To this end, two main approaches have been proposed: (i) models based on *local node aggregation*, usually built upon Graph Neural Networks (GNNs) (Kipf and Welling, 2017; Hamilton et al., 2017) and (ii) models that leverage *global node aggregation*. Systems that adopt global encoding strategies are typically based on Transformers (Vaswani et al. 2017), using self-attention to compute a node representation based on all nodes in the graph. This approach enjoys the advantage of a large node context range, but neglects the graph topology by effectively treating every node as being connected to all the others in the graph. In contrast, models based on local aggregation learn the representation of each node based on its adjacent nodes as defined in the input graph. This approach effectively exploits the graph topology, and the graph structure has a strong impact on the node representation (Xu et al., 2018). However, encoding relations between distant nodes can be challenging by requiring more graph encoding layers, which can also propagate noise (Li et al., 2018).

For example, Figure 1a presents a KG, for which a corresponding text is shown in Figure 1b. Note that there is a mismatch between how entities are connected in the graph and how their natural language descriptions are related in the text. Some entities syntactically related in the text are not connected in the graph. For instance, in the

[1]Code is available at https://github.com/UKPLab/kg2text.

Transactions of the Association for Computational Linguistics, vol. 8, pp. 589–604, 2020. https://doi.org/10.1162/tacl_a_00332
Action Editor: Alessandro Moschitti. Submission batch: 2/2019; Revision batch: 5/2020; Published 9/2020.

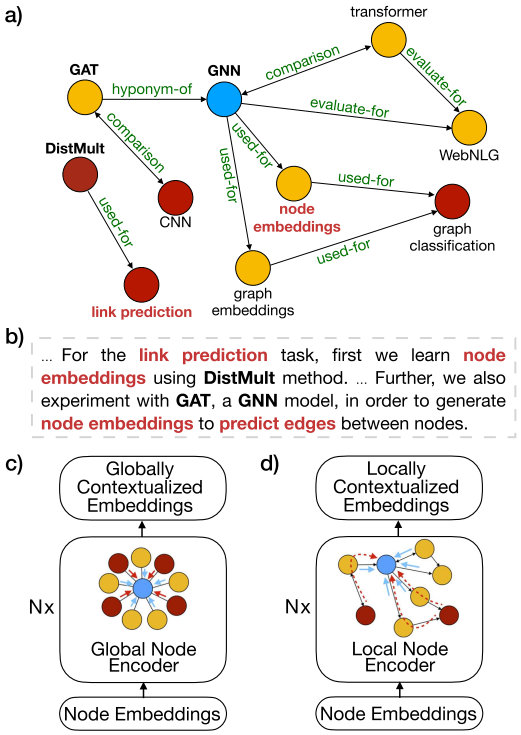


Figure 1: A graphical representation (a) of a scientific text (b). (c) A global encoder directly captures longer dependencies between any pair of nodes (blue and red arrows), but fails in capturing the graph structure. (d) A local encoder explicitly accesses information from the adjacent nodes (blue arrows) and implicitly captures distant information (dashed red arrows).

sentence ''For *the link prediction task*, first we learn *node embeddings* using *DistMult method*.'', although the entity mentions are dependent of the same verb, in the graph, the *node embeddings* node has no explicit connection with *link prediction* and *DistMult* nodes, which are in a different connected component. This example illustrates the importance of encoding distant information in the input graph. As shown in Figure 1c, a global encoder is able to learn a node representation for *node embeddings* which captures information from non-connected entities such as *DistMult*. By modeling distant connections between all nodes, we allow for these missing links to be captured, as KGs are known to be highly incomplete (Dong et al., 2014; Schlichtkrull et al., 2018).

In contrast, the local strategy refines the node representation with richer neighborhood information, as nodes that share the same neighborhood exhibit a strong *homophily*: Two similar entities are much more likely to be connected than at random. Consequently, the local context enriches the node representation with local information from KG triples. For example, in Figure 1a, *GAT* reaches *node embeddings* through the *GNN*. This transitive relation can be captured by a local encoder, as shown in Figure 1d. Capturing this form of relationship also can support text generation at the sentence level.

In this paper, we investigate novel graph-to-text architectures that combine both *global* and *local* node aggregations, gathering the benefits from both strategies. In particular, we propose a unified graph-to-text framework based on Graph Attention Networks (GATs) (Veličković et al., 2018). As part of this framework, we empirically compare two main architectures: a *cascaded architecture* that performs global node aggregation before performing local node aggregation, and a *parallel architecture* that performs global and local aggregations simultaneously. The cascaded architecture allows the local encoder to leverage global encoding features, and the parallel architecture allows more independent features to complement each other. To further consider fine-grained integration, we additionally consider *layer-wise integration* of the global and local encoders.

Extensive experiments show that our approaches consistently outperform recent models on two benchmarks for text generation from KGs. To the best of our knowledge, we are the first to consider integrating global and local context aggregation in graph-to-text generation, and the first to propose a unified GAT structure for combining global and local node contexts.

2 Related Work

Early efforts for graph-to-text generation used statistical methods (Flanigan et al., 2016; Pourdamghani et al., 2016; Song et al., 2017). Recently, several neural graph-to-text models have exhibited success by leveraging encoder mechanisms based on LSTMs, GNNs, and Transformers.

AMR-to-Text Generation. Various neural models have been proposed to generate sentences from Abstract Meaning Representation (AMR) graphs. Konstas et al. (2017) provide the first neural approach for this task, by linearizing the input graph as a sequence of nodes and edges. Song et al. (2018) propose the graph recurrent network to directly encode the AMR nodes, whereas Beck et al. (2018) develop a model based on gated GNNs.

However, both approaches only use local node aggregation strategies. Damonte and Cohen (2019) combine graph convolutional networks and LSTMs in order to learn complementary node contexts. However, differently from Transformers and GNNs, LSTMs generate node representations that are influenced by the node order. Ribeiro et al. (2019) develop a model based on different GNNs that learns node representations which simultaneously encode a top–down and a bottom–up views of the AMR graphs, whereas Guo et al. (2019) leverage dense connectivity in GNNs. Recently, Wang et al. (2020) propose a local graph encoder based on Transformers using separated attentions for incoming and outgoing neighbors. Recent methods (Zhu et al., 2019; Cai and Lam, 2020) also use Transformers, but learn globalized node representations, modeling graph paths in order to capture structural relations.

KG-to-Text Generation. In this work, we focus on generating text from KGs. In comparison to AMRs, which are rooted and connected graphs, KGs do not have a defined topology, which may vary widely among different datasets, making the generation process more demanding. KGs are sparse structures that potentially contain a large number of relations. Moreover, we are typically interested in generating multisentence texts from KGs, and this involves solving document planning issues (Konstas and Lapata, 2013).

Recent neural approaches for KG-to-text generation simply linearize the KG triples, thereby loosing graph structure information. For instance, Colin and Gardent (2018), Moryossef et al. (2019), and Adapt (Gardent et al., 2017) utilize LSTM/GRU to encode WebNLG graphs. Castro Ferreira et al. (2019) systematically compare pipeline and end-to-end models for text generation from WebNLG graphs. Trisedya et al. (2018) develop a graph encoder based on LSTMs that captures relationships within and between triples. Previous work has also studied how to explicitly encode the graph structure using GNNs or Transformers. Marcheggiani and Perez Beltrachini (2018) propose an encoder based on graph convolutional networks, that consider explicitly local node contexts, and show superior performance compared with LSTMs. Recently, Koncel-Kedziorski et al. (2019) proposed a Transformer-based approach that computes the node representations by attending over node neighborhoods following a self-attention strategy. In contrast, our models focus on distinct global and local message passing mechanisms, capturing complementary graph contexts.

Integrating Global Information. There has been recent work that attempts to integrate global context in order to learn better node representations in graph-to-text generation. To this end, existing methods use an artificial global node for message exchange with the other nodes. This strategy can be regarded as extending the graph structure but using similar message passing mechanisms. In particular, Koncel-Kedziorski et al. (2019) add a global node to the graph and use its representation to initialize the decoder. Recently, Guo et al. (2019) and Cai and Lam (2020) also utilized an artificial global node with direct edges to all other nodes to allow global message exchange for AMR-to-text generation. Similarly, Zhang et al. (2018) use a global node to a graph recurrent network model for sentence representation. Different from the above methods, we consider integrating global and local contexts at the *node* level, rather than the *graph* level, by investigating *model* alternatives rather than *graph structure* changes. In addition, we integrate GAT and Transformer architectures into a unified global-local model.

3 Graph-to-Text Model

This section first describes (i) the graph transformation adopted to create a relational graph from the input (Section 3.1), and (ii) the graph encoders of our framework based on GAT (Veličković et al., 2018), for dealing with both global (Section 3.3) and local (Section 3.4) node contexts. We adopt GAT because it is closely related to the Transformer architecture (Vaswani et al., 2017), which provides a convenient prototype for modeling global node context. Then, (iii) we proposed strategies to combined the global and local graph enoders (Section 3.5). Finally, (iv) we describe the decoding and training procedures (Section 3.6).

3.1 Graph Preparation

We represent a KG as a multi-relational graph[2] $\mathcal{G}_e = (\mathcal{V}_e, \mathcal{E}_e, \mathcal{R})$ with entity nodes $e \in \mathcal{V}_e$ and labeled edges $(e_h, r, e_t) \in \mathcal{E}_e$, where $r \in \mathcal{R}$

[2]In this paper, multi-relational graphs refer to directed graphs with labeled edges.

denotes the relation existing from the entity e_h to e_t.[3]

Unlike other current approaches (Koncel-Kedziorski et al., 2019; Moryossef et al., 2019), we represent an entity as a set of nodes. For instance, the KG node "node embedding" in Figure 1 will be represented by two nodes, one for the token "node" and the other for the token "embedding". Formally, we transform each $\mathcal{G}_e$ into a new graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$, where each token of an entity $e \in \mathcal{V}_e$ becomes a node $v \in \mathcal{V}$. We convert each edge $(e_h, r, e_t) \in \mathcal{E}_e$ into a set of edges (with the same relation r) and connect every token of e_h to every token of e_t. That is, an edge (u, r, v) will belong to $\mathcal{E}$ if and only if there exists an edge $(e_h, r, e_t) \in \mathcal{E}_e$ such that $u \in e_h$ and $v \in e_t$, where e_h and e_t are seen as sets of tokens. We represent each node $v \in \mathcal{V}$ with an embedding $h_v^0 \in \mathbb{R}^{d_v}$, generated from its corresponding token.

The new graph $\mathcal{G}$ increases the representational power of the models because it allows learning node embeddings at a token level, instead of entity level. This is particularly important for text generation as it permits the model to be more flexible, capturing richer relationships between entity tokens. This also allows the model to learn relations and attention functions between source and target tokens. However, it has the side effect of removing the natural sequential order of multi-word entities. To preserve this information, we use position embeddings (Vaswani et al., 2017), that is, h_v^0 becomes the sum of the corresponding token embedding and the positional embedding for v.

3.2 Graph Neural Networks (GNN)

Multilayer GNNs work by iteratively learning a representation vector h_v of a node v based on both its context node neighbors and edge features, through an information propagation scheme. More formally, the l-th layer aggregates the representations of v's context nodes:

$$h_{\mathcal{N}(v)}^{(l)} = \text{AGGR}^{(l)}\left(\left\{\left(h_u^{(l-1)}, r_{uv}\right) : u \in \mathcal{N}(v)\right\}\right),$$

where $\text{AGGR}^{(l)}(.)$ is an aggregation function, shared by all nodes on the l-th layer. r_{uv} represents the relation between u and v. $\mathcal{N}(v)$ is a set of context nodes for v. In most GNNs, the context nodes are those adjacent to v. $h_{\mathcal{N}(v)}^{(l)}$ is the aggregated context representation of $\mathcal{N}(v)$ at layer l. $h_{\mathcal{N}(v)}^{(l)}$ is used to update the representation of v:

$$h_v^{(l)} = \text{COMBINE}^{(l)}\left(h_v^{(l-1)}, h_{\mathcal{N}(v)}^{(l)}\right).$$

After L iterations, a node's representation encodes the structural information within its L-hop neighborhood. The choices of $\text{AGGR}^{(l)}(.)$ and $\text{COMBINE}^{(l)}(.)$ differ by the specific GNN model. An example of $\text{AGGR}^{(l)}(.)$ is the sum of the representations of $\mathcal{N}(v)$. An example of $\text{COMBINE}^{(l)}(.)$ is a concatenation after the feature transformation.

3.3 Global Graph Encoder

A global graph encoder aggregates the *global context* for updating each node based on all nodes of the graph (see Figure 1c). We use the attention mechanism as the message passing scheme, extending the self-attention network structure of Transformer to a GAT structure. In particular, we compute a layer of the *global convolution* for a node $v \in \mathcal{V}$, which takes the input feature representations h_v as input, adopting $\text{AGGR}^{(l)}(.)$ as:

$$h_{\mathcal{N}(v)} = \sum\nolimits_{u \in \mathcal{V}} \alpha_{vu} W_g\, h_u, \tag{1}$$

where $W_g \in \mathbb{R}^{d_v \times d_z}$ is a model parameter. The attention weight α_{vu} is calculated as:

$$\alpha_{vu} = \frac{\exp(e_{vu})}{\sum_{k \in \mathcal{V}} \exp(e_{vk})}, \tag{2}$$

where

$$e_{vu} = \left(\left(W_q h_v\right)^\top \left(W_k h_u\right)\right) / d_z \tag{3}$$

is the attention function which measures the *global importance* of node u's features to node v. $W_q, W_k \in \mathbb{R}^{d_v \times d_z}$ are model parameters and d_z is a scaling factor. To capture distinct relations between nodes, K independent global convolutions are calculated and concatenated:

$$\hat{h}_{\mathcal{N}(v)} = \Big\|_{k=1}^{K} h_{\mathcal{N}(v)}^{(k)}. \tag{4}$$

Finally, we define $\text{COMBINE}^{(l)}(.)$ using layer normalization (LayerNorm) and a fully connected

[3] $\mathcal{R}$ contains relations both in canonical direction (e.g., used-for) and in inverse direction (e.g., used-for-inv), so that the models consider the differences in the incoming and outgoing relations.

feed-forward network (FFN), in a similar way as the transformer architecture:

$$\hat{h}_v = \text{LayerNorm}(\hat{h}_{\mathcal{N}(v)} + h_v), \quad (5)$$

$$h_v^{global} = \text{FFN}(\hat{h}_v) + \hat{h}_{\mathcal{N}(v)} + h_v. \quad (6)$$

Note that the global encoder creates an artificial complete graph with $\mathcal{O}(n^2)$ edges and does not consider the edge relations. In particular, if the labeled edges were considered, the self-attention space complexity would increase to $\Theta(|\mathcal{R}|n^2)$.

3.4 Local Graph Encoder

The representation h_v^{global} captures macro relationships from v to all other nodes in the graph. However, this representation lacks both structural information regarding the local neighborhood of v and the graph topology. Also, it does not capture labeled edges (relations) between nodes (see Equations 1 and 3). In order to capture these crucial graph properties and impose a strong relational inductive bias, we build a graph encoder to aggregate the *local context* by utilizing a modified version of GAT augmented with relational weights. In particular, we compute a layer of the *local convolution* for a node $v \in \mathcal{V}$, adopting $\text{AGGR}^{(l)}(.)$ as:

$$h_{\mathcal{N}(v)} = \sum\nolimits_{u \in \mathcal{N}(v)} \alpha_{vu} W_r h_u, \quad (7)$$

where $W_r \in \mathbb{R}^{d_v \times d_z}$ encodes the relation $r \in \mathcal{R}$ between u and v. $\mathcal{N}(v)$ is a set of nodes adjacent to v and v itself. The attention coefficient α_{vu} is computed as:

$$\alpha_{vu} = \frac{\exp(e_{vu})}{\sum_{k \in \mathcal{N}(v)} \exp(e_{vk})}, \quad (8)$$

where

$$e_{vu} = \sigma\left(a^\top [W_v h_v \parallel W_r h_u]\right) \quad (9)$$

is the attention function which calculates the *local importance* of adjacent nodes, considering the edge labels. σ is an activation function, $\parallel$ denotes concatenation and $W_v \in \mathbb{R}^{d_v \times d_z}$ and $a \in \mathbb{R}^{2d_z}$ are model parameters.

We use multihead attentions to learn local relations in different perspectives, as in Equation 4, generating $\hat{h}_{\mathcal{N}(v)}$. Finally, we define $\text{COMBINE}^{(l)}(.)$ as:

$$h_v^{local} = \text{RNN}(h_v, \hat{h}_{\mathcal{N}(v)}), \quad (10)$$

where we use as RNN a Gated Recurrent Unit (GRU) (Cho et al., 2014). GRU facilitates information propagation between local layers. This choice is motivated by recent work (Xu et al., 2018; Dehmamy et al., 2019) that theoretically demonstrates that sharing information between layers helps the structural signals propagate. In a similar direction, AMR-to-text generation models use LSTMs (Song et al., 2017) and dense connections (Guo et al., 2019) between GNN layers.

3.5 Combining Global and Local Encodings

Our goal is to implement a graph encoder capable of encoding global and local aspects of the input graph. We hypothesize that these two sources of information are complementary, and a combination of both enriches node representations for text generation. In order to test this hypothesis, we investigate different combined architectures.

Intuitively, there are two general methods for integrating two types of representation. The first is to concatenate vectors of global and local contexts, which we call a *parallel* representation. The second is to form a pipeline, where a global representation is first obtained, which is then used as a input for calculating refined representations based on the local node context. We call this approach a *cascaded* representation.

Parallel and cascaded integration can be performed at the model level, considering the global and local graph encoders as two representation learning units disregarding internal structures. However, because our model takes a multilayer architecture, where each layer makes a level of abstraction in representation, we can alternatively consider integration on the layer level, so that more interaction between global and local contexts may be captured. As a result, we present four architectures for integration, as shown in Figure 2. All models serve the same purpose, and their relative strengths should be evaluated empirically.

Parallel Graph Encoding (PGE). In this setup, we compose global and local graph encoders in a fully parallel structure (Figure 2a). Note that each graph encoder can have different numbers of layers and attention heads. The final node

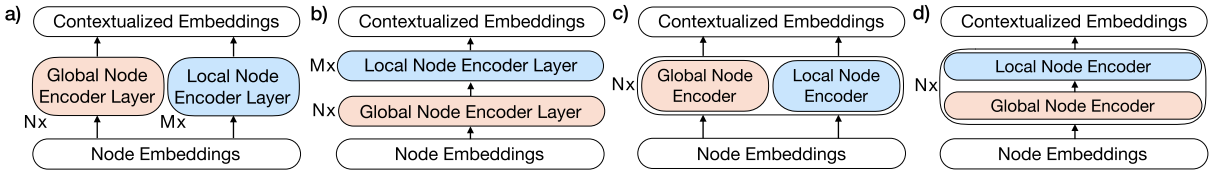


Figure 2: Overview of the proposed encoder architectures. (a) Parallel Graph Encoder (PGE) with separated parallel global and local node encoders. (b) Cascaded Graph Encoder (CGE) with separated cascaded encoders. (c) PGE−LW: global and local node representations are concatenated layer-wise. (d) CGE−LW: Both node representations are cascaded layer-wise.

representation is the concatenation of the local and global node representations of the last layers of both graph encoders:

$$\begin{aligned} h_v^{global} &= \text{GE}(h_v^0, \{h_u^0 : u \in \mathcal{V}\}) \\ h_v^{local} &= \text{LE}(h_v^0, \{h_u^0 : u \in \mathcal{N}(v)\}) \\ h_v &= [\, h_v^{global} \parallel h_v^{local} \,], \end{aligned} \tag{11}$$

where GE and LE denote the global and local graph encoders, respectively. h_v^0 is the initial node embedding used in the first layer of both encoders.

Cascaded Graph Encoding (CGE). We cascade local and global graph encoders as shown in Figure 2b. We first compute a globally contextualized node embedding, and then refine it with the local node context. h_v^0 is the initial input for the global encoder and h_v^{global} is the initial input for the local encoder. In particular, the final node representation is calculated as follows:

$$\begin{aligned} h_v^{global} &= \text{GE}(h_v^0, \{h_u^0 : u \in \mathcal{V}\}) \\ h_v &= \text{LE}(h_v^{global}, \{h_u^{global} : u \in \mathcal{N}(v)\}). \end{aligned} \tag{12}$$

Layer-wise Parallel Graph Encoding. To allow fine-grained interaction between the two types of graph contextual information, we also combine the encoders in a layer-wise (LW) fashion. As shown in Figure 2c, for each graph layer, we use both global and local encoders in a parallel structure (PGE−LW). More precisely, each encoder layer is calculates as follows:

$$\begin{aligned} h_v^{global} &= \text{GE}_l(h_v^{l-1}, \{h_u^{l-1} : u \in \mathcal{V}\}) \\ h_v^{local} &= \text{LE}_l(h_v^{l-1}, \{h_u^{l-1} : u \in \mathcal{N}(v)\}) \\ h_v^l &= [\, h_v^{global} \parallel h_v^{local} \,], \end{aligned} \tag{13}$$

where GE_l and LE_l refer to the l-th layers of the global and local graph encoders, respectively.

Layer-wise Cascaded Graph Encoding. We also propose cascading the graph encoders layer-wise (CGE−LW, Figure 2d). In particular, we compute each encoder layer as follows:

$$\begin{aligned} h_v^{global} &= \text{GE}_l(h_v^{l-1}, \{h_u^{l-1} : u \in \mathcal{V}\}) \\ h_v^l &= \text{LE}_l(h_v^{global}, \{h_u^{global} : u \in \mathcal{N}(v)\}). \end{aligned} \tag{14}$$

3.6 Decoder and Training

Our decoder follows the core architecture of a Transformer decoder (Vaswani et al., 2017). Each time step t is updated by performing multihead attentions over the output of the encoder (node embeddings h_v) and over previously generated tokens (token embeddings). An additional challenge in our setup is to generate multisentence outputs. In order to encourage the model to generate longer texts, we implement a length penalty (Wu et al., 2016) to refine the pure max-probability beam search.

The model is trained to optimize the negative log-likelihood of each gold-standard output text. We use label smoothing regularization to prevent the model from predicting the tokens too confidently during training and generalizing poorly.

4 Data and Preprocessing

We attest the effectiveness of our models on two datasets: AGENDA (Koncel-Kedziorski et al., 2019) and WebNLG (Gardent et al., 2017). Table 1 shows the statistics for both datasets.

AGENDA. In this dataset, KGs are paired with scientific abstracts extracted from proceedings of 12 top AI conferences. Each instance consists of the paper title, a KG, and the paper abstract. Entities correspond to scientific terms that are often multiword expressions (co-referential entities are merged). We treat each token in the title as a node, creating a unique graph with title and KG

	#train	#dev	#test	#relations	avg #entities	avg #nodes	avg #edges	avg #CC	avg length
AGENDA	38,720	1,000	1,000	7	12.4	44.3	68.6	19.1	140.3
WebNLG	18,102	872	971	373	4.0	34.9	101.0	1.5	24.2

Table 1: Data statistics. Nodes, edges, and CC values are calculated after the graph transformation. The average values are calculated for all splits (training, dev, and test sets). CC refers to the number of connected components.

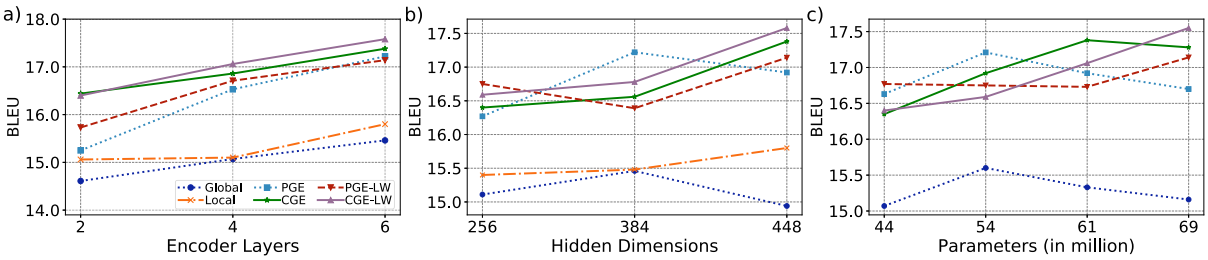


Figure 3: BLEU scores for the AGENDA dev set, with respect to (a) the encoder layers, (b) the encoder hidden dimensions, and (c) the number of parameters.

tokens as nodes. As shown in Table 1, the average output length is considerably large, as the target outputs are multisentence abstracts.

WebNLG. In this dataset, each instance contains a KG extracted from DBPedia. The target text consists of sentences that verbalize the graph. We evaluate the models on the test set with seen categories. Note that this dataset has a considerable number of edge relations (see Table 1). In order to avoid parameter explosion, we use regularization based on the basis function decomposition to define the model relation weights (Schlichtkrull et al., 2018). Also, as an alternative, we use the Levi Transformation to create nodes from relational edges between entities (Beck et al., 2018). That is, we create a new relation node for each edge relation between two nodes. The new relation node is connected to the subject and object token entities by two binary relations, respectively.

5 Experiments

We implemented all our models using PyTorch Geometric (PyG) (Fey and Lenssen, 2019) and OpenNMT-py (Klein et al., 2017). We use the Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.98$. Our learning rate schedule follows Vaswani et al. (2017), with 8,000 and 16,000 warming-up steps for WebNLG and AGENDA, respectively. The vocabulary is shared between the node and target tokens. In order to mitigate the effects of random seeds, for the test sets, we report the averages over 4 training runs along with their standard deviation. We use byte pair encoding (Sennrich et al., 2016) to split entity words into smaller more frequent pieces. Therefore some nodes in the graph can be sub-words. We also obtain sub-words on the target side. Following previous works, we evaluate the results with BLEU (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2014), and CHRF++ (Popović, 2015) automatic metrics and also perform a human evaluation (Section 5.6). For layer-wise models, the number of encoder layers are chosen from $\{2, 4, 6\}$, and for `PGE` and `CGE`, the global and local layers are chosen from and $\{2, 4, 6\}$ and $\{1, 2, 3\}$, respectively. The hidden encoder dimensions are chosen from $\{256, 384, 448\}$ (see Figure 3). Hyperparameters are tuned on the development set of both datasets. We report the test results when the BLEU score on dev set is optimal.

5.1 Results on AGENDA

Table 2 shows the results, where we report the number of layers and attention heads utilized. We train models with only global or local encoders as baselines. Each model has the respective parameter size that gives the best results on the dev set. First, the local encoder, which requires fewer encoder layers and parameters, has a better performance compared with the global encoder. This shows that explicitly encoding the graph structure

Model	#L	#H	BLEU	METEOR	CHRF++	#P
Koncel-Kedziorski et al. (2019)	6	8	14.30 ± 1.01	18.80 ± 0.28	–	–
Global Encoder	6	8	15.44 ± 0.25	20.76 ± 0.194	43.95 ± 0.40	54.4
Local Encoder	3	8	16.03 ± 0.19	21.12 ± 0.32	44.70 ± 0.29	54.0
PGE	6, 3	8, 8	17.55 ± 0.154	22.02 ± 0.07	**46.41** ± 0.07	56.1
CGE	6, 3	8, 8	**17.82** ± 0.134	**22.23** ± 0.09	**46.47** ± 0.10	61.5
PGE-LW	6	8, 8	17.42 ± 0.25	21.78 ± 0.20	45.79 ± 0.32	69.0
CGE-LW	6	8, 8	**18.01** ± 0.14	**22.34** ± 0.07	**46.69** ± 0.17	69.8

Table 2: Results on the AGENDA test set. #L and #H are the numbers of layers and the attention heads in each layer, respectively. When more than one, the values are for the global and local encoders, respectively. #P stands for the number of parameters in millions (node embeddings included).

Model	BLEU	METEOR	CHRF++	#P
UPF-FORGe (Gardent et al., 2017)	40.88	40.00	–	–
Melbourne (Gardent et al., 2017)	54.52	41.00	70.72	–
Adapt (Gardent et al., 2017)	60.59	44.00	76.01	–
Marcheggiani and Perez Beltrachini (2018)	55.90	39.00	–	4.9
Trisedya et al. (2018)	58.60	40.60	–	–
Castro Ferreira et al. (2019)	57.20	41.00	–	–
CGE	62.30 ± 0.27	43.51 ± 0.18	75.49 ± 0.34	13.9
CGE (Levi Graph)	63.10 ± 0.13	44.11 ± 0.09	76.33 ± 0.10	12.8
CGE-LW	62.85 ± 0.07	43.75 ± 0.21	75.73 ± 0.31	11.2
CGE-LW (Levi Graph)	**63.69** ± 0.10	**44.47** ± 0.12	**76.66** ± 0.10	10.4

Table 3: Results on the WebNLG test set with seen categories.

is important to improve the node representations. Second, our approaches substantially outperform both baselines. CGE−LW outperforms Koncel-Kedziorski et al. (2019), a transformer model that focuses on the relations between adjacent nodes, by a large margin, achieving the new state-of-the-art BLEU score of 18.01, 25.9% higher. We also note that KGs are highly incomplete in this dataset, with an average number of connected components of 19.1 (see Table 1). For this reason, the global encoder plays an important role in our models as it enables learning node representations based on all connected components. The results indicate that combining the local node context, leveraging the graph topology, and the global node context, capturing macro-level node relations, leads to better performance. We find that, even though CGE has a small number of parameters compared to CGE−LW, it achieves comparable performance. PGE−LW has the worse performance among the proposed models. Finally, note that cascaded architectures are more effective according to different metrics.

5.2 Results on WebNLG

We compare the performance of our more effective models (CGE, CGE−LW) with six state-of-the-art results reported on this dataset. Three systems are the best competitors in the WebNLG challenge for seen categories: UPF-FORGe, Melbourne, and Adapt. UPF-FORGe follows a rule-based approach, whereas the others use neural encoder-decoder models with linearized triple sets as input.

Table 3 presents the results. CGE achieves a BLEU score of 62.30, 8.9% better than the best model of Castro Ferreira et al. (2019), who use an end-to-end architecture based on GRUs. CGE using Levi graphs outperforms Trisedya et al. (2018), an approach that encodes both intra-triple and inter-triple relationships, by 4.5 BLEU points. Interestingly, their intra-triple and inter-triple mechanisms are closely related with the local and global encodings. However, they rely on encoding entities based on sequences generated by traversal graph algorithms, whereas we explicitly

exploit the graph structure, throughout the local neighborhood aggregation.

CGE-LW with Levi graphs as inputs has the best performance, achieving 63.69 BLEU points, even thought it uses fewer parameters. Note that this approach allows the model to handle new relations, as they are treated as nodes. Moreover, the relations become part of the shared vocabulary, making this information directly usable during the decoding phase. We outperform an approach based on GNNs (Marcheggiani and Perez Beltrachini, 2018) by a large margin of 7.7 BLEU points, showing that our combined graph encoding strategies lead to better text generation. We also outperform Adapt, a strong competitor that utilizes subword encodings, by 3.1 BLEU points.

5.3 Development Experiments

We report several development experiments in Figure 3. Figure 3a shows the effect of the number of encoder layers in the four encoding methods.[4] In general, the performance increases when we gradually enlarge the number of layers, achieving the best performance with 6 encoder layers. Figure 3b shows the choices of hidden sizes for the encoders. The best performances for global and PGE are achieved with 384 dimensions, whereas the other models have the better performance with 448 dimensions. In Figure 3c, we evaluate the performance employing different number of parameters.[5] When the models are smaller, parallel encoders obtain better results than the cascaded ones. When the models are larger, cascaded models perform better. We speculate that for some models, the performance can be further improved with more parameters and layers. However, we do not attempt this owing to hardware limitations.

5.4 Ablation Study

In Table 4, we report an ablation study on the impact of each module used in CGE model on the dev set of AGENDA. We also report the number of parameters used in each configuration.

Global Graph Encoder. We start by an ablation on the global encoder. After removing the global attention coefficients, the performance of the model drops by 1.79 BLEU and 1.97 CHRF++ scores. Results also show that using FFN in the global COMBINE(.) function is important to the model but less effective than the global attention. However, when we remove FNN, the number of parameters drops considerably (around 18%) from 61.5 to 50.4 million. Finally, without the entire global encoder, the result drops substantially by 2.21 BLEU points. This indicates that enriching node embeddings with a global context allows learning more expressive graph representations.

Model	BLEU	CHRF++	#P
CGE	17.38	45.68	61.5
Global Encoder			
-Global Attention	15.59	43.71	59.0
-FFN	16.33	44.86	50.4
-Global Encoder	15.17	43.30	45.6
Local Encoder			
-Local Attention	16.92	45.97	61.5
-Weight Relations	16.88	45.61	53.6
-GRU	16.38	44.71	60.2
-Local Encoder	14.68	42.98	51.8
-Shared Vocab.	16.92	46.16	81.8
Decoder			
–Length Penalty	16.68	44.68	61.5

Table 4: Ablation study for modules used in the encoder and decoder of the CGE model.

Local Graph Encoder. We first remove the local graph attention and the BLEU score drops to 16.92, showing that the neighborhood attention improves the performance. After removing the relation types, encoded as model weights, the performance drops by 0.5 BLEU points. However, the number of parameters is reduced by around 7.9 million. This indicates that we can have a more efficient model, in terms of the number of parameters, with a slight drop in performance. Removing the GRU used on the COMBINE(.) function decreases the performance considerably. The worse performance occurs if we remove the entire local encoder, with a BLEU score of 14.68, essentially making the encoder similar to the global baseline.

Finally, we find that vocabulary sharing improves the performance, and the length penalty is beneficial as we generate multisentence outputs.

[4]For CGE and PGE the values refer to the global layers and the number of local layers is fixed to 3.

[5]It was not possible to execute the local model with larger number of parameters because of memory limitations.

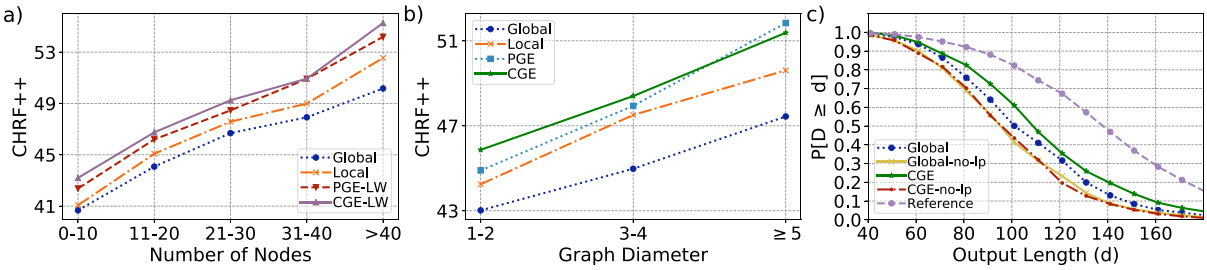


Figure 4: CHRF++ scores for the AGENDA test set, with respect to (a) the number of nodes, and (b) the graph diameter. (c) Distribution of length of the gold references and models' outputs for the AGENDA test set.

5.5 Impact of the Graph Structure and Output Length

The overall performance on both datasets suggests the strength of combining global and local node representations. However, we are also interested in estimating the models' performance concerning different data properties.

Graph Size. Figure 4a shows the effect of the graph size, measured in number of nodes, on the performance, measured using CHRF++ scores,[6] for the AGENDA. We evaluate global and local graph encoders, `PGE-LW` and `CGE-LW`. We find that the score increases as the graph size increases. Interestingly, the gap between the local and global encoders increases when the graph size increases. This suggests that, because larger graphs may have very different topologies, modeling the relations between nodes based on the graph structure is more beneficial than allowing direct communication between nodes, overlooking the graph structure. Also note that the the cascaded model (`CGE-LW`) is consistently better than the parallel model (`PGE-LW`) over all graph sizes.

Table 5 shows the effect of the graph size, measured in number of triples, on the performance for the WebNLG. Our model obtains better scores over all partitions. In contrast to AGENDA, the performance decreases as the graph size increases. This behavior highlights a crucial difference between AGENDA and AMR and WebNLG datasets, in which the models' general performance decreases as the graph size increases (Gardent et al., 2017; Cai and Lam, 2020). In WebNLG, the graph and sentence sizes are correlated, and longer sentences are more challenging to generate than the smaller ones. Differently, AGENDA contains similar text lengths[7] and when the input is a larger graph, the model has more information to be leveraged during the generation.

Graph Diameter. Figure 4b shows the impact of the graph diameter[8] on the performance for the AGENDA. Similarly to the graph size, the score increases as the diameter increases. As the global encoder is not aware of the graph structure, this module has the worst scores, even though it enables direct node communication over long distance. In contrast, the local encoder can propagate precise node information throughout the graph structure for k-hop distances, making the relative performance better. Table 5 shows the models' performances with respect to the graph diameter for WebNLG. Similarly to the graph size, the score decreases as the diameter increases.

Output Length. One interesting phenomenon to analyze is the length distribution (in number of words) of the generated outputs. We expect that our models generate texts with similar output lengths as the reference texts. As shown in Figure 4c, the references usually are bigger than the texts generated by all models for AGENDA. The texts generated by `CGE-no-pl`, a `CGE` model without length penalty, are consistently shorter than the texts from the global and `CGE` models. We increase the length of the texts when we use the length penalty (see Section 3.6). However, there is still a gap between the reference and the generated text lengths. We leave further investigation of this aspect for future work.

[6] CHRF++ score is used as it is a sentence-level metric.

[7] As shown on Figure 4c, 82% of the reference abstracts have more than 100 words.

[8] The diameter of a graph is defined as the length of the longest shortest path between two nodes. We convert the graphs into undirected graphs to calculate the diameters.

#T	#DP	Melbourne	Adapt	CGE-LW
1-2	396	78.74	83.10	84.35
3-4	386	66.84	72.02	72.27
5-7	189	61.85	69.28	70.25
#D	**#DP**	**Melbourne**	**Adapt**	**CGE-LW**
1	222	82.27	87.54	88.04
2	469	69.94	74.54	75.90
≥ 3	280	62.87	69.30	69.41
#S	**#DP**	**Melbourne**	**Adapt**	**CGE-LW**
1	388	77.19	81.66	82.03
2	306	67.29	73.29	73.78
3	151	66.30	72.46	73.21
4	66	66.73	71.26	75.16
≥ 5	60	61.93	67.57	69.20

Table 5: CHRF++ scores with respect to the number of triples (#T), graph diameters (#D), and number of sentences (#S) on the WebNLG test set. #DP refers to the number of datapoints.

Table 5 shows the models' performances with respect to the number of sentences for WebNLG. In general, increasing the number of sentences reduces the performance of all models. Note that when the number of sentences increases, the gap between CGE-LW and the baselines becomes larger. This suggests that our approach is able to better handle complex graph inputs in order to generate multisentence texts.

Effect of the Number of Nodes on the Output Length. Figure 5 shows the effect of the size of a graph, defined as the number of nodes, on the quality (measured in CHRF++ scores) and length of the generated text (in number of words) in the AGENDA dev set. We bin both the graph size and the output length in 4 classes. CGE consistently outperforms the global model, in some cases by a large margin. When handling smaller graphs (with ≤ 35 nodes), both models have difficulties generating good summaries. However, for these smaller graphs, our model achieves a score 12.2% better when generating texts with length ≤ 75. Interestingly, when generating longer texts (>140) from smaller graphs, our model outperforms the global encoder by an impressive 21.7%, indicating that our model is more effective in capturing semantic signals from graphs with scarce information. Our approach also performs better when the graph size is large (> 55) but the generation output is small (≤ 75), beating the global encoder by 9 points.

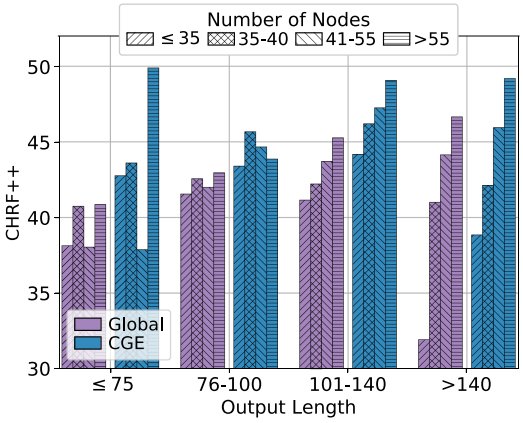


Figure 5: Relation between the number of nodes and the length of the generated text, in number of words.

5.6 Human Evaluation

To further assess the quality of the generated text, we conduct a human evaluation on the WebNLG dataset.[9] Following previous work (Gardent et al., 2017; Castro Ferreira et al., 2019), we assess two quality criteria: (i) *Fluency* (i.e., does the text flow in a natural, easy to read manner?) and (ii) *Adequacy* (i.e., does the text clearly express the data?). We divide the datapoints into seven different sets by the number of triples. For each set, we randomly select 20 texts generated by Adapt, CGE with Levi graphs, and their corresponding human reference (420 texts in total). Because the number of datapoints for each set is not balanced (see Table 5), this sampling strategy ensures that we have the same number of samples for the different triple sets. Moreover, having human references may serve as an indicator of the sanity of the human evaluation experiment. We recruited human workers from Amazon Mechanical Turk to rate the text outputs on a 1–5 Likert scale. For each text, we collect scores from 4 workers and average them. Table 6 shows the results. We first note a similar trend as in the automatic evaluation, with CGE outperforming Adapt on both fluency and adequacy. In sets with the number of triples smaller than 5, CGE was the highest rated system in fluency. Similarly to the automatic evaluation, both systems are better in generating text from

[9]Because AGENDA is scientific in nature, we choose to crowd-source human evaluations only for WebNLG.

#T	Adapt		CGE		Reference	
	F	A	F	A	F	A
All	3.96^{C}	4.44^{C}	4.12^{B}	4.54^{B}	4.24^{A}	4.63^{A}
1–2	3.94^{C}	4.59^{B}	4.18^{B}	4.72^{A}	4.30^{A}	4.69^{A}
3–4	3.79^{C}	4.45^{B}	3.96^{B}	4.50^{AB}	4.14^{A}	4.66^{A}
5–7	4.08^{B}	4.35^{B}	4.18^{B}	4.45^{B}	4.28^{A}	4.59^{A}
#D	**Adapt**		**CGE**		**Reference**	
	F	A	F	A	F	A
1–2	3.98^{C}	4.50^{B}	4.16^{B}	4.61^{A}	4.28^{A}	4.66^{A}
≥ 3	3.91^{C}	4.33^{B}	4.03^{B}	4.43^{B}	4.17^{A}	4.60^{A}

Table 6: Fluency (F) and Adequacy (A) obtained in the human evaluation. #T refers to the number of input triples and #D to graph diameters. The ranking was determined by pair-wise Mann-Whitney tests with $p < 0.05$, and the difference between systems that have a letter in common is not statistically significant.

AGENDA		
Model	**BLEU**	**CHRF++**
CGE-LW	18.17	46.80
-Shared Vocab	17.88	47.12
-Length Penalty	17.46	45.76
-Both	17.24	46.14
WebNLG		
Model	**BLEU**	**CHRF++**
CGE-LW	63.86	76.80
-Shared Vocab	63.07	76.17
-Length Penalty	63.28	76.51
-Both	62.60	75.80

Table 7: Effects of the vocabulary sharing and length penalty on the test sets of AGENDA and WebNLG.

graphs with smaller diameters. Note that bigger diameters pose difficulties to the models, which achieve their worst performance for diameters ≥ 3.

5.7 Additional Experiments

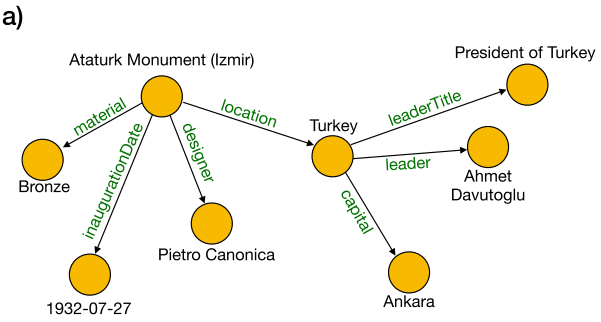
Impact of the Vocabulary Sharing and Length Penalty. During the ablation studies, we note that the vocabulary sharing and length penalty are beneficial for the performance. To better estimate their impact, we evaluate `CGE-LW` model with its variations without using vocabulary sharing, length penalty and without both mechanisms, on the test set of both datasets. Table 7 shows the results. We observe that sharing vocabulary is more important to WebNLG than AGENDA. This suggests that sharing vocabulary is beneficial when the training data is small, as in WebNLG. On the other hand, length penalty is more effective for AGENDA, as it has longer texts than WebNLG,[10] improving the BLEU score by 0.71 points.

How Far Does the Global Attention Look? Following previous work (Voita et al., 2019; Cai and Lam, 2020), we investigate the attention distribution of each graph encoder global layer of `CGE-LW` on the AGENDA dev set. In particular, for each node, we verify its global neighbor that receives the maximum attention weight and record the distance between them.[11] Figure 7 shows the averaged distances for each global layer. We observe that the global encoder mainly focuses on distant nodes, instead of the neighbors and closest nodes. This is very interesting and agrees with our intuition: Whereas the local encoder is concerned about the local neighborhood, the global encoder focuses on the information from long-distance nodes.

Case Study. Figure 6 shows examples of generated texts when the WebNLG graph is complex (7 triples). While `CGE` generates a factually correct text (it correctly verbalises all triples), the Adapt's output is repetitive. The example also illustrates how the text generated by `CGE` closely follows the graph structure whereby the first sentence verbalises the right-most subgraph, the second the left-most one and the linking node *Turkey* makes the transition (using hyperonymy and a definite description, i.e., *The country*). The text created by `CGE` is also more coherent than the reference. As noted above, the input graph includes two subgraphs linked by *Turkey*. In natural language, such a meaning representation corresponds to a topic shift with the first part of the text describing an entity from one subgraph, the second part an entity from the other subgraph, and the linking entity (*Turkey*) marking the topic shift. Typically,

[10] As shown in Table 1, AGENDA has texts 5.8 times longer than WebNLG on average.

[11] The distance between two nodes is defined as the number of edges in a shortest path connecting them.



b) Adapt: Ahmet Davutoglu is the president of Turkey. The capital city is Ankara, but it is in Izmir that the bronze Ataturk monument designed by Pietro Canonica and inaugurated on 27 July 1932 is located. The monument was designed in bronze by Pietro Canonica and inaugurated on 27 July 1932.

c) CGE: President Ahmet Davutoglu is the leader of Turkey where the capital city is Ankara. The country is the location of the bronze Ataturk monument designed by Pietro Canonica and inaugurated on 27 July 1932 in Izmir.

d) Reference: President Ahmet Davutoglu is the Turkish leader where the capital city is Ankara. The bronze Ataturk Monument which was designed by Pietro Canonica is located in Izmir and was inaugurated on 27 July 1932.

Figure 6: (a) A WebNLG input graph and the outputs for (b) Adapt and (c) CGE. The colored text indicates repetition.

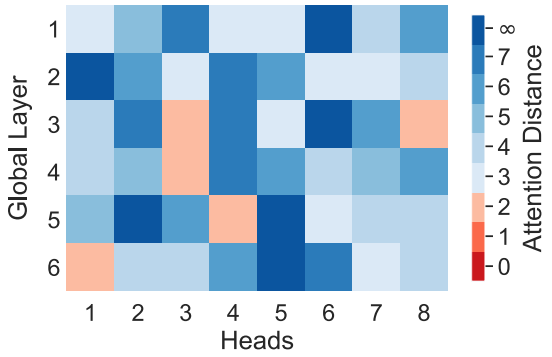


Figure 7: The average distance between nodes for the maximum attention for each head. ∞ indicates no path between two nodes, that is, they belong to distinct connected components.

in English, a topic shift is marked by a definite noun phrase in the subject position. Although this is precisely the discourse structure generated by CGE (*Turkey* is realized in the second sentence by the definite description *The country* in the subject position), the reference fails to mark the topic shift, resulting in a text with weaker discourse coherence.

6 Conclusion

In this work, we introduced a unified graph attention network structure for investigating graph-to-text models that combines global and local graph encoders in order to improve text generation. An extensive evaluation of our models demonstrated that the global and local contexts are empirically complementary, and a combination can achieve state-of-the-art results on two datasets. In addition, cascaded architectures give better results compared with parallel ones.

We point out some directions for future work. First, it is interesting to study different fusion strategies to assemble the global and local encodings. Second, a promising direction is incorporating pre-trained contextualized word embeddings in graphs. Third, as discussed in Section 5.5, it is worth studying ways to diminish the gap between the reference and the generated text lengths.

Acknowledgments

We would like to thank Pedro Savarese, Markus Zopf, Mohsen Mesgar, Prasetya Ajie Utama, Ji-Ung Lee, and Kevin Stowe for their feedback on this work, as well as the anonymous reviewers for detailed comments that improved this paper. This work has been supported by the German Research Foundation as part of the Research Training Group Adaptive Preparation of Information from Heterogeneous Sources (AIPHES) under grant No. GRK 1994/1.

References

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

Deng Cai and Wai Lam. 2020. Graph transformer for graph-to-sequence learning. In *Proceedings of The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*.

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

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Doha, Qatar. Association for Computational Linguistics.

Emilie Colin and Claire Gardent. 2018. Generating syntactic paraphrases. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 937–943, Brussels, Belgium. Association for Computational Linguistics.

Marco Damonte and Shay B. Cohen. 2019. Structural neural encoders for AMR-to-text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3649–3658. Minneapolis, Minnesota. Association for Computational Linguistics.

Nima Dehmamy, Albert-Laszlo Barabasi, and Rose Yu. 2019. Understanding the representation power of graph neural networks in learning graph topology, In H. Wallach, H. Larochelle, A. Beygelzimer, F. Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 15387–15397. Curran Associates, Inc.

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

Xin Luna Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,z KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 601–610.

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

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

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

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

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs, In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 1024–1034. Curran Associates, Inc.

Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th*

International Conference on Learning Representations, ICLR '17.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72. Vancouver, Canada. Association for Computational Linguistics.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text Generation from Knowledge Graphs with Graph Transformers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2284–2293, Minneapolis, Minnesota. Association for Computational Linguistics.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural amr: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157, Vancouver, Canada. Association for Computational Linguistics.

Ioannis Konstas and Mirella Lapata. 2013. Inducing document plans for concept-to-text generation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1503–1514, Seattle, Washington, USA. Association for Computational Linguistics.

Q. Li, Z. Han, and X.-M. Wu. 2018. Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning. In *The Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI.

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

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019. Step-by-step: Separating planning from realization in neural data-to-text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2267–2277, Minneapolis, Minnesota. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318. Stroudsburg, PA, USA. Association for Computational Linguistics.

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

Nima Pourdamghani, Kevin Knight, and Ulf Hermjakob. 2016. Generating English from abstract meaning representations. In *Proceedings of the 9th International Natural Language Generation conference*, pages 21–25, Edinburgh, UK. Association for Computational Linguistics.

Leonardo F. R. Ribeiro, Claire Gardent, and Iryna Gurevych. 2019. Enhancing AMR-to-text generation with dual graph representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3181–3192, Hong Kong, China. Association for Computational Linguistics.

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

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of*

the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Linfeng Song, Xiaochang Peng, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2017. AMR-to-text generation with synchronous node replacement grammar. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 7–13, Vancouver, Canada. Association for Computational Linguistics.

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

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

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

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Tianming Wang, Xiaojun Wan, and Hanqi Jin. 2020. AMR-to-text generation with graph transformer. *Transactions of the Association for Computational Linguistics*, 8:19–33.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation learning on graphs with jumping knowledge networks. In *ICML*.

Yue Zhang, Qi Liu, and Linfeng Song. 2018. Sentence-state LSTM for text representation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 317–327, Melbourne, Australia. Association for Computational Linguistics.

Jie Zhu, Junhui Li, Muhua Zhu, Longhua Qian, Min Zhang, and Guodong Zhou. 2019. Modeling graph structure in transformer for better AMR-to-text generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5458–5467, Hong Kong, China. Association for Computational Linguistics.

Chapter 6

Modeling Graph Structure via Relative Position for Text Generation from Knowledge Graphs

Modeling Graph Structure via Relative Position for Text Generation from Knowledge Graphs

Martin Schmitt[1] **Leonardo F. R. Ribeiro**[2] **Philipp Dufter**[1] **Iryna Gurevych**[2] **Hinrich Schütze**[1]

[1]Center for Information and Language Processing (CIS), LMU Munich
[2]Research Training Group AIPHES and UKP Lab, Technische Universität Darmstadt
martin@cis.lmu.de

Abstract

We present *Graformer*, a novel Transformer-based encoder-decoder architecture for graph-to-text generation. With our novel graph self-attention, the encoding of a node relies on all nodes in the input graph – not only direct neighbors – facilitating the detection of global patterns. We represent the relation between two nodes as the length of the shortest path between them. Graformer learns to weight these node-node relations differently for different attention heads, thus virtually learning differently connected views of the input graph. We evaluate Graformer on two popular graph-to-text generation benchmarks, AGENDA and WebNLG, where it achieves strong performance while using many fewer parameters than other approaches.[1]

1 Introduction

A knowledge graph (KG) is a flexible data structure commonly used to store both general world knowledge (Auer et al., 2008) and specialized information, e.g., in biomedicine (Wishart et al., 2018) and computer vision (Krishna et al., 2017). Generating a natural language description of such a graph (KG→text) makes the stored information accessible to a broader audience of end users. It is therefore important for KG-based question answering (Bhowmik and de Melo, 2018), data-to-document generation (Moryossef et al., 2019; Koncel-Kedziorski et al., 2019) and interpretability of KGs in general (Schmitt et al., 2020).

Recent approaches to KG→text employ encoder-decoder architectures: the encoder first computes vector representations of the graph's nodes, the decoder then uses them to predict the text sequence. Typical encoder choices are graph neural networks based on message passing between direct neighbors in the graph (Kipf and Welling, 2017; Veličković et al., 2018) or variants of Transformer (Vaswani et al., 2017) that apply self-attention on all nodes together, including those that are not directly connected. To avoid losing information, the latter approaches use edge or node *labels* from the shortest path when computing the attention between two nodes (Zhu et al., 2019; Cai and Lam, 2020). Assuming the existence of a path between any two nodes is particularly problematic for KGs: a set of KG facts often does not form a connected graph.

We propose a flexible alternative that neither needs such an assumption nor uses label information to model graph structure: a Transformer-based encoder that interprets the *lengths* of shortest paths in a graph as relative position information and thus, by means of multi-head attention, dynamically learns different structural views of the input graph with differently weighted connection patterns. We call this new architecture *Graformer*.

Following previous work, we evaluate Graformer on two benchmarks: (i) the AGENDA dataset (Koncel-Kedziorski et al., 2019), i.e., the generation of scientific abstracts from automatically extracted entities and relations specific to scientific text, and (ii) the WebNLG challenge dataset (Gardent et al., 2017), i.e., the task of generating text from DBPedia subgraphs. On both datasets, Graformer achieves more than 96% of the state-of-the-art performance while using only about half as many parameters.

In summary, our contributions are as follows: (1) We develop *Graformer*, a novel graph-to-text architecture that interprets shortest path lengths as relative position information in a graph self-attention network. (2) Graformer achieves competitive performance on two popular KG-to-text generation benchmarks, showing that our architecture can learn about graph structure without any guidance other than its text generation objective. (3) To further investigate what Graformer learns about graph structure, we visualize the differently connected

[1]Our code is publicly available: https://github.com/mnschmit/graformer

Proceedings of the Fifteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-15), pages 10–21

graph views it has learned and indeed find different attention heads for more local and more global graph information. Interestingly, direct neighbors are considered particularly important even without any structural bias, such as introduced by a graph neural network. (4) Analyzing the performance w.r.t. different input graph properties, we find evidence that Graformer's more elaborate global view on the graph is an advantage when it is important to distinguish between distant but connected nodes and truly unreachable ones.

2 Related Work

Most recent approaches to graph-to-text generation employ a graph neural network (GNN) based on message passing through the input graph's topology as the encoder in their encoder-decoder architectures (Marcheggiani and Perez-Beltrachini, 2018; Koncel-Kedziorski et al., 2019; Ribeiro et al., 2019; Guo et al., 2019). As one layer of these encoders only considers immediate neighbors, a large number of stacked layers can be necessary to learn about distant nodes, which in turn also increases the risk of propagating noise (Li et al., 2018).

Other approaches (Zhu et al., 2019; Cai and Lam, 2020) base their encoder on the Transformer architecture (Vaswani et al., 2017) and thus, in each layer, compute self-attention on all nodes, not only direct neighbors, facilitating the information flow between distant nodes. Like Graformer, these approaches incorporate information about the graph topology with some variant of relative position embeddings (Shaw et al., 2018). They, however, assume that there is always a path between any pair of nodes, i.e., there are no unreachable nodes or disconnected subgraphs. Thus they use an LSTM (Hochreiter and Schmidhuber, 1997) to compute a relation embedding from the labels along this path. However, in contrast to the AMR[2] graphs used for their evaluation, KGs are frequently disconnected. Graformer is more flexible and makes no assumption about connectivity. Furthermore, its relative position embeddings only depend on the *lengths* of shortest paths i.e., purely structural information, not labels. It thus effectively learns *differently connected views* of its input graph.

Deficiencies in modeling long-range dependencies in GNNs have been considered a serious limitation before. Various solutions orthogonal to our approach have been proposed in recent work: By incorporating a connectivity score into their graph attention network, Zhang et al. (2020) manage to increase the attention span to k-hop neighborhoods but, finally, only experiment with $k = 2$. Our graph encoder efficiently handles dependencies between much more distant nodes. Pei et al. (2020) define an additional neighborhood based on Euclidean distance in a continuous node embedding space. Similar to our work, a node can thus receive information from distant nodes, given their embeddings are close enough. However, Pei et al. (2020) compute these embeddings only once before training whereas in our approach node similarity is based on the learned representation in each encoder layer. This allows Graformer to dynamically change node interaction patterns during training.

Recently, Ribeiro et al. (2020) use two GNN encoders – one using the original topology and one with a fully connected version of the graph – and combine their output in various ways for graph-to-text generation. This approach can only see two extreme versions of the graph: direct neighbors and full connection. Our approach is more flexible and dynamically learns a different structural view per attention head. It is also more parameter-efficient as our multi-view encoder does not need a separate set of parameters for each view.

3 The Graformer Model

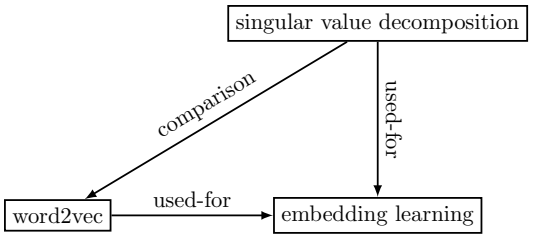
Graformer follows the general multi-layer encoder-decoder pattern known from the original Transformer (Vaswani et al., 2017). In the following, we first describe our formalization of the KG input and then how it is processed by Graformer.

3.1 Graph data structure

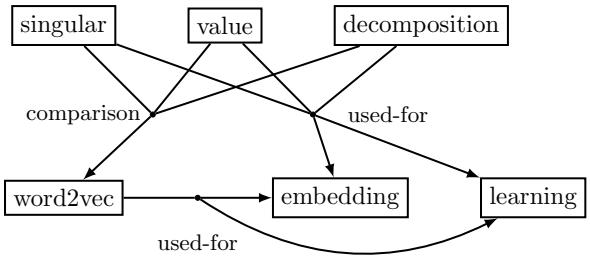
Knowledge graph. We formalize a knowledge graph (KG) as a directed, labeled multigraph $G_{KG} = (V, A, s, t, l_V, l_A, \mathcal{E}, \mathcal{R})$ with V a set of vertices (the KG entities), A a set of arcs (the KG facts), $s, t : A \to V$ functions assigning to each arc its source/target node (the subject/object of a KG fact), and $l_V : V \to \mathcal{E}, l_A : A \to \mathcal{R}$ providing labels for vertices and arcs, where $\mathcal{R}$ is a set of KG-specific relations and $\mathcal{E}$ a set of entity names.

Token graph. Entity names usually consist of more than one token or subword unit. Hence, a tokenizer $tok : \mathcal{E} \to \Sigma_T^*$ is needed that splits an entity's label into its components from the vocabulary Σ_T of text tokens. Following recent work (Ribeiro et al., 2020), we mimic this composition-

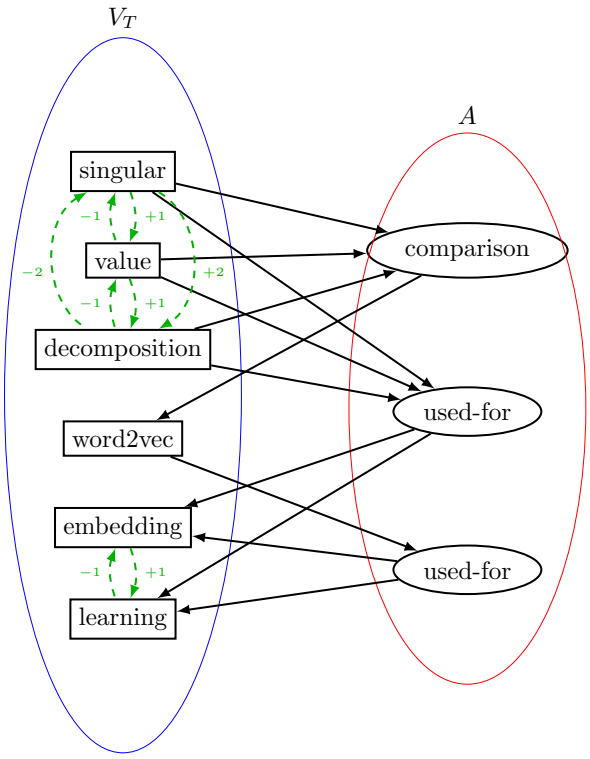
[2] abstract meaning representation



(a) Original knowledge graph



(b) Directed hypergraph (token graph)



(c) Incidence graph with SAME_p edges (dashed green)

Figure 1: Different representations of the same KG (types are omitted for clarity).

ality of node labels in the graph structure by splitting each node into as many nodes as there are tokens in its label. We thus obtain a directed hypergraph $G_T = (V_T, A, s_T, t_T, l_T, l_A, \Sigma_T, \mathcal{R}, same)$, where $s_T, t_T : A \to \mathcal{P}(V_T)$ now assign a set of source (resp. target) nodes to each (hyper-) arc and all nodes are labeled with only one token, i.e., $l_T : V_T \to \Sigma_T$. Unlike Ribeiro et al. (2020), we additionally keep track of all token nodes' origins: $same : V_T \to \mathcal{P}(V_T \times \mathbb{Z})$ assigns to each node n all other nodes n' stemming from the same entity together with the relative position of $l_T(n)$ and $l_T(n')$ in the original tokenized entity name. Fig. 1b shows the token graph corresponding to the KG in Fig. 1a.

Incidence graph. For ease of implementation, our final data structure for the KG is the hypergraph's incidence graph, a bipartite graph where hyper-arcs are represented as nodes and edges are unlabeled: $G = (N, E, l, \Sigma, \{\, \text{SAME}_p \mid p \in \mathbb{Z} \,\})$ where $N = V_T \cup A$ is the set of nodes, $E = \{\, (n_1, n_2) \mid n_1 \in s_T(n_2) \vee n_2 \in t_T(n_1) \,\}$ the set of directed edges, $l : N \to \Sigma$ a label function, and $\Sigma = \Sigma_T \cup \mathcal{R}$ the vocabulary. We introduce SAME_p edges to fully connect $same$ clusters: $\text{SAME}_p = \{\, (n_1, n_2) \mid (n_2, p) \in same(n_1) \,\}$ where p differentiates between different relative positions in the original entity string, similar to (Shaw et al., 2018). See Fig. 1c for an example.

3.2 Graformer encoder

The initial graph representation $\boldsymbol{H}^{(0)} \in \mathbb{R}^{|N| \times d}$ is obtained by looking up embeddings for the node labels in the learned embedding matrix $\boldsymbol{E} \in \mathbb{R}^{|\Sigma| \times d}$, i.e., $\boldsymbol{H}_i^{(0)} = \boldsymbol{e}^{l(n_i)} \boldsymbol{E}$ where $\boldsymbol{e}^{l(n_i)}$ is the one-hot-encoding of the ith node's label.

To compute the node representation $\boldsymbol{H}^{(L)}$ in the Lth layer, we follow Vaswani et al. (2017), i.e., we first normalize the input from the previous layer $\boldsymbol{H}^{(L-1)}$ via layer normalization LN, followed by multi-head graph self-attention $SelfAtt_g$ (see § 3.3 for details), which – after dropout regularization Dr and a residual connection – yields the intermediate representation $\boldsymbol{\mathcal{I}}$ (cf. Eq. (1)). A feedforward layer FF with one hidden layer and GeLU (Hendrycks and Gimpel, 2016) activation computes the final layer output (cf. Eq. (2)). As recommended by Chen et al. (2018), we apply an additional layer normalization step to the output $\boldsymbol{H}^{(L_E)}$ of the last encoder layer L_E.

$$\boldsymbol{\mathcal{I}}^{(L)} = Dr(SelfAtt_g(LN(\boldsymbol{H}^{(L-1)}))) + \boldsymbol{H}^{(L-1)} \tag{1}$$

$$\boldsymbol{H}^{(L)} = Dr(FF(LN(\boldsymbol{\mathcal{I}}^{(L)}))) + \boldsymbol{\mathcal{I}}^{(L)} \tag{2}$$

$SelfAtt_g$ computes a weighted sum of $\boldsymbol{H}^{(L-1)}$:

$$SelfAtt_g(\boldsymbol{H})_i = \sum_{j=1}^{|N|} \alpha_{ij}^g (\boldsymbol{H}_j \boldsymbol{W}^{V_g}) \tag{3}$$

where $\boldsymbol{W}^{V_g} \in \mathbb{R}^{d \times d}$ is a learned parameter matrix.

In the next section, we derive the definition of the graph-structure-informed attention weights α_{ij}^g.

3.3 Self-attention for text and graphs with relative position embeddings

In this section, we describe the computation of attention weights for multi-head self-attention. Note that the formulas describe the computations for one head. The output of multiple heads is combined as in the original Transformer (Vaswani et al., 2017).

Text self-attention. Shaw et al. (2018) introduced position-aware self-attention in the Transformer by (i) adding a relative position embedding $\mathbf{A}^K \in \mathbb{R}^{M\times M\times d}$ to $\boldsymbol{X}$'s key representation, when computing the softmax-normalized attention scores $\boldsymbol{\alpha}_i$ between $\boldsymbol{X}_i \in \mathbb{R}^d$ and the complete input embedding matrix $\boldsymbol{X} \in \mathbb{R}^{M\times d}$ (cf. Eq. (4)), and (ii) adding a second type of position embedding $\mathbf{A}^V \in \mathbb{R}^{M\times M\times d}$ to $\boldsymbol{X}$'s value representation when computing the weighted sum (cf. Eq. (5)):

$$\boldsymbol{\alpha}_i = \sigma\left(\frac{\boldsymbol{X}_i\boldsymbol{W}^Q(\boldsymbol{X}\boldsymbol{W}^K + \mathbf{A}_i^K)^\top}{\sqrt{d}}\right) \quad (4)$$

$$\boldsymbol{V}_i = \sum_{j=1}^{n}\alpha_{ij}(\boldsymbol{X}_j\boldsymbol{W}^V + \mathbf{A}_{ij}^V) \quad (5)$$

where $\sigma(\cdot)$ denotes the softmax function, i.e.,

$$\sigma(\boldsymbol{b})_i = \frac{\exp(b_i)}{\sum_{j=1}^J \exp(b_j)}, \quad \text{for } \boldsymbol{b} \in \mathbb{R}^J.$$

Recent work (Raffel et al., 2019) has adopted a simplified form where value-modifying embeddings $\mathbf{A}^V$ are omitted and key-modifying embeddings $\mathbf{A}^K$ are replaced with learned scalar embeddings $\boldsymbol{S} \in \mathbb{R}^{M\times M}$ that – based on relative position – directly in- or decrease attention scores before normalization, i.e., Eq. (4) becomes Eq. (6).

$$\boldsymbol{\alpha}_i = \sigma\left(\frac{\boldsymbol{X}_i\boldsymbol{W}^Q(\boldsymbol{X}\boldsymbol{W}^K)^\top}{\sqrt{d}} + \boldsymbol{S}_i\right) \quad (6)$$

Shaw et al. (2018) share their position embeddings across attention heads but learn separate embeddings for each layer as word representations from different layers can vary a lot. Raffel et al. (2019) learn separate $\boldsymbol{S}$ matrices for each attention head but share them across layers. We use Raffel et al. (2019)'s form of relative position encoding for text self-attention in our decoder (§ 3.4).

Graph self-attention. Analogously to self-attention on text, we define our structural graph self-attention as follows:

	V_T						A		
	s	v	d	w	e	l	c	u1	u2
s	0	4	5	2	2	2	1	1	3
v	-4	0	4	2	2	2	1	1	3
d	-5	-4	0	2	2	2	1	1	3
w	-2	-2	-2	0	2	2	-1	∞	1
e	-2	-2	-2	-2	0	4	-3	-1	-1
l	-2	-2	-2	-2	-4	0	-3	-1	-1
c	-1	-1	-1	1	3	3	0	∞	2
u1	-1	-1	-1	∞	1	1	∞	0	∞
u2	-3	-3	-3	-1	1	1	-2	∞	0

Figure 2: $\boldsymbol{R}$ matrix for the graph in Fig. 1c ($\delta_{max} = 3$).

$$\boldsymbol{\alpha}_i^g = \sigma\left(\frac{\boldsymbol{H}_i\boldsymbol{W}^{Q_g}(\boldsymbol{H}\boldsymbol{W}^{K_g})^\top}{\sqrt{d}} + \gamma(\boldsymbol{R})_i\right) \quad (7)$$

$\boldsymbol{W}^{K_g}, \boldsymbol{W}^{Q_g} \in \mathbb{R}^{d\times d}$ are learned matrices and $\gamma : \mathbb{Z}\cup\{\infty\} \rightarrow \mathbb{R}$ looks up *learned scalar embeddings* for the relative graph positions in $\boldsymbol{R} \in \mathbb{R}^{N\times N}$.

We define the relative graph position R_{ij} between the nodes n_i and n_j with respect to two factors: (i) the text relative position p in the original entity name if n_i and n_j stem from the same original entity, i.e., $(n_i, n_j) \in \text{SAME}_p$ for some p and (ii) shortest path lengths otherwise:

$$R_{ij} = \begin{cases} \infty, & \text{if } \delta(n_i,n_j) = \infty \\ & \text{and } \delta(n_j,n_i) = \infty \\ encode(p), & \text{if } (n_i,n_j) \in \text{SAME}_p \\ \delta(n_i,n_j), & \text{if } \delta(n_i,n_j) \le \delta(n_j,n_i) \\ -\delta(n_j,n_i), & \text{if } \delta(n_i,n_j) > \delta(n_j,n_i) \end{cases} \quad (8)$$

where $\delta(n_i, n_j)$ is the length of the shortest path from n_i to n_j, which we define to be ∞ if and only if there is no such path. *encode* maps a text relative position $p \in \mathbb{Z}\setminus\{0\}$ to an integer outside δ's range to avoid clashes. Concretely, we use $encode(p) := sgn(p)\cdot\delta_{max} + p$ where δ_{max} is the maximum graph diameter, i.e., the maximum value of δ over all graphs under consideration.

Thus, we model graph relative position as the length of the shortest path using either only forward edges ($R_{ij} > 0$) or only backward edges ($R_{ij} < 0$). Additionally, two special cases are considered: (i) Nodes without any purely forward or purely backward path between them ($R_{ij} = \infty$) and (ii) token nodes from the same entity. Here the relative position in the original entity string p is encoded outside the range of path length encodings (which are always in the interval $[-\delta_{max}, \delta_{max}]$).

In practice, we use two thresholds, n_δ and n_p. All values of δ exceeding n_δ are set to n_δ and analogously for p. This limits the number of different positions a model can distinguish.

Intuition. Our definition of relative position in graphs combines several advantages: (i) Any node can attend to any other node – even unreachable ones – while learning a suitable attention bias for different distances. (ii) SAME_p edges are treated differently in the attention mechanism. Thus, entity representations can be learned like in a regular transformer encoder, given that tokens from the same entity are fully connected with SAME_p edges with p providing relative position information. (iii) The lengths of shortest paths often have an intuitively useful interpretation in our incidence graphs and the sign of the entries in $\boldsymbol{R}$ also captures the important distinction between incoming and outgoing paths. In this way, Graformer can, e.g., capture the difference between the subject and object of a fact, which is expressed as a relative position of -1 vs. 1. The subject and object nodes, in turn, see each other as 2 and -2, respectively.

Fig. 2 shows the $\boldsymbol{R}$ matrix corresponding to the graph from Fig. 1c. Note how token nodes from the same entity, e.g., `s`, `v`, and `d`, form clusters as they have the same distances to other nodes, and how the relations inside such a cluster are encoded outside the interval $[-3, 3]$, i.e., the range of shortest path lengths. It is also insightful to compare node pairs with the same value in $\boldsymbol{R}$. E.g., both `s` and `w` see `e` at a distance of 2 because the entities *SVD* and *word2vec* are both the subject of a fact with *embedding learning* as the object. Likewise, `s` sees both `c` and `u1` at a distance of 1 because its entity *SVD* is subject to both corresponding facts.

3.4 Graformer decoder

Our decoder follows closely the standard Transformer decoder (Vaswani et al., 2017), except for the modifications suggested by Chen et al. (2018).

Hidden decoder representation. The initial decoder representation $\boldsymbol{Z}^{(0)} \in \mathbb{R}^{M \times d}$ embeds the (partially generated) target text $\boldsymbol{T} \in \mathbb{R}^{M \times |\Sigma|}$, i.e., $\boldsymbol{Z}^{(0)} = \boldsymbol{T}\boldsymbol{E}$. A decoder layer L then obtains a contextualized representation via self-attention as in the encoder (§ 3.2):

$$\boldsymbol{C}^{(L)} = Dr(SelfAtt_t(LN(\boldsymbol{Z}^{(L-1)}))) + \boldsymbol{Z}^{(L-1)} \tag{9}$$

$SelfAtt_t$ differs from $SelfAtt_g$ by using different position embeddings in Eq. (7) and, obviously, R_{ij} is defined in the usual way for text. $\boldsymbol{C}^{(L)}$ is then modified via multi-head attention MHA on the output $\boldsymbol{H}^{(L_E)}$ of the last graph encoder layer L_E. As in § 3.2, we make use of residual connections, layer normalization LN, and dropout Dr:

$$\boldsymbol{U}^{(L)} = Dr(MHA(LN(\boldsymbol{C}^{(L)}), \boldsymbol{H}^{(L_E)})) + \boldsymbol{C}^{(L)} \tag{10}$$

$$\boldsymbol{Z}^{(L)} = Dr(FF(LN(\boldsymbol{U}^{(L)}))) + \boldsymbol{U}^{(L)} \tag{11}$$

where

$$MHA(\boldsymbol{C}, \boldsymbol{H})_i = \sum_{j=1}^{|N|} \alpha_{ij}(\boldsymbol{H}_j \boldsymbol{W}^{V_t}) \tag{12}$$

$$\boldsymbol{\alpha}_i = \sigma\left(\frac{\boldsymbol{C}_i \boldsymbol{W}^{Q_t} (\boldsymbol{H}\boldsymbol{W}^{K_t})^\top}{\sqrt{d}}\right) \tag{13}$$

Generation probabilities. The final representation $\boldsymbol{Z}^{(L_D)}$ of the last decoder layer L_D is used to compute the probability distribution $\boldsymbol{P}_i \in [0, 1]^{|\Sigma|}$ over all words in the vocabulary Σ at time step i:

$$\boldsymbol{P}_i = \sigma\left(\boldsymbol{Z}_i^{(L_D)} \boldsymbol{E}^\top\right) \tag{14}$$

Note that $\boldsymbol{E} \in \mathbb{R}^{|\Sigma| \times d}$ is the same matrix that is also used to embed node labels and text tokens.

3.5 Training

We train Graformer by minimizing the standard negative log-likelihood loss based on the likelihood estimations described in the previous section.

4 Experiments

4.1 Datasets

We evaluate our new architecture on two popular benchmarks for KG-to-text generation, AGENDA (Koncel-Kedziorski et al., 2019) and WebNLG (Gardent et al., 2017). While the latter contains crowd-sourced texts corresponding to subgraphs from various DBPedia categories, the former was automatically created by applying an information extraction tool (Luan et al., 2018) on a corpus of scientific abstracts (Ammar et al., 2018). As this process is noisy, we corrected 7 train instances where an entity name was erroneously split on a special character and, for the same reason, deleted 1 train instance entirely. Otherwise, we use the data as is, including the train/dev/test split.

We list the number of instances per data split, as well as general statistics about the graphs in Table 1. Note that the graphs in WebNLG are human-authored subgraphs of DBpedia while the graphs

	AGENDA	WebNLG
#instances in train	38,719	18,102
#instances in val	1,000	872
#instances in test	1,000	971
#relation types	7	373
avg #entities in KG	13.4	4.0
% connected graphs	0.3	99.9
avg #graph components	8.4	1.0
avg component size	1.6	3.9
avg #token nodes in graph	98.0	36.0
avg #tokens in text	157.9	31.5
avg % text tokens in graph	42.7	56.1
avg % graph tokens in text	48.6	49.0
Vocabulary size $\|\Sigma\|$	24,100	2,100
Character coverage in %	99.99	100.0

Table 1: Statistics of AGENDA and the dataset from the WebNLG challenge as used in our experiments. Upper part: data splits and original KGs. Lower part: token graphs and BPE settings.

in AGENDA were automatically extracted. This leads to a higher number of disconnected graph components. Nearly all WebNLG graphs consist of a single component, i.e., are connected graphs, whereas for AGENDA this is practically never the case. We also report statistics that depend on the tokenization (cf. § 4.2) as factors like the length of target texts and the percentage of tokens shared verbatim between input graph and target text largely impact the task difficulty.

4.2 Data preprocessing

Following previous work on AGENDA (Ribeiro et al., 2020), we put the paper title into the graph as another entity. In contrast to Ribeiro et al. (2020), we also link every node from a real entity to every node from the title by TITLE2TXT and TXT2TITLE edges. The type information provided by AGENDA is, as usual for KGs, expressed with one dedicated node per type and HAS-TYPE arcs that link entities to their types. We keep the original pretokenized texts but lowercase the title as both node labels and target texts are also lowercased.

For WebNLG, we follow previous work (Gardent et al., 2017) by replacing underscores in entity names with whitespace and breaking apart camelcased relations. We furthermore follow the evaluation protocol of the original challenge by converting all characters to lowercased ASCII and separating all punctuation from alphanumeric characters during tokenization.

For both datasets, we train a BPE vocabulary using sentencepiece (Kudo and Richardson, 2018) on the train set, i.e., a concatenation of node labels and target texts. See Table 1 for vocabulary sizes. Note that for AGENDA, only 99.99% of the characters found in the train set are added to the vocabulary. This excludes exotic Unicode characters that occur in certain abstracts.

We prepend entity and relation labels with dedicated $\langle E \rangle$ and $\langle R \rangle$ tags.

4.3 Hyperparameters and training details

We train Graformer with the Adafactor optimizer (Shazeer and Stern, 2018) for 40 epochs on AGENDA and 200 epochs on WebNLG. We report test results for the model yielding the best validation performance measured in corpus-level BLEU (Papineni et al., 2002). For model selection, we decode greedily. The final results are generated by beam search. Following Ribeiro et al. (2020), we couple beam search with a length penalty (Wu et al., 2016) of 5.0. See Appendix A for more details and a full list of hyperparameters.

4.4 Epoch curriculum

We apply a data loading scheme inspired by the bucketing approach of Koncel-Kedziorski et al. (2019) and length-based curriculum learning (Platanios et al., 2019): We sort the train set by target text length and split it into four buckets of two times 40% and two times 10% of the data. After each training epoch, the buckets are shuffled internally but their global order stays the same from shorter target texts to longer ones. This reduces padding during batching as texts of similar lengths stay together and introduces a mini-curriculum from presumably easier examples (i.e., shorter targets) to more difficult ones for each epoch. This enables us to successfully train Graformer *even without a learning rate schedule*.

5 Results and Discussion

5.1 Overall performance

Table 2 shows the results of our evaluation on AGENDA in terms of BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), and CHRF++ (Popović, 2017). Like the models we compare with, we report the average and standard deviation of 4 runs with different random seeds.

Our model outperforms previous Transformer-based models that only consider first-order neighborhoods per encoder layer (Koncel-Kedziorski et al., 2019; An et al., 2019). Compared to the very

	BLEU	METEOR	CHRF++	#P
Ours	17.80 ±0.31	22.07 ±0.23	45.43 ±0.39	36.3
GT	14.30 ±1.01	18.80 ±0.28	–	–
GT+RBS	15.1 ±0.97	19.5 ±0.29	–	–
CGE-LW	18.01 ±0.14	22.34 ±0.07	46.69 ±0.17	69.8

Table 2: Experimental results on AGENDA. GT (Graph Transformer) from (Koncel-Kedziorski et al., 2019); GT+RBS from (An et al., 2019); CGE-LW from (Ribeiro et al., 2020). Number of parameters in millions.

	BLEU	METEOR	CHRF++	#P
Ours	61.15 ±0.22	43.38 ±0.17	75.43 ±0.19	5.3
UPF-FORGe	40.88	40.00	–	–
Melbourne	54.52	41.00	70.72	–
Adapt	60.59	44.00	76.01	–
Graph Conv.	55.90	39.00	–	4.9
GTR-LSTM	58.60	40.60	–	–
E2E GRU	57.20	41.00	–	–
CGE-LW-LG	63.69 ±0.10	44.47 ±0.12	76.66 ±0.10	10.4

Table 3: Experimental results on the WebNLG test set with seen categories. CGE-LW-LG from (Ribeiro et al., 2020); Adapt, Melbourne and UPF-FORGe from (Gardent et al., 2017); Graph Conv. from (Marcheggiani and Perez-Beltrachini, 2018); GTR-LSTM from (Trisedya et al., 2018); E2E GRU from (Castro Ferreira et al., 2019). Number of parameters in millions.

recent models by Ribeiro et al. (2020), Graformer performs very similarly. Using both a local and a global graph encoder, Ribeiro et al. (2020) combine information from very distant nodes but at the same time need extra parameters for the second encoder. Graformer is more efficient and still matches their best model's BLEU and METEOR scores within a standard deviation.

The results on the test set of seen categories of WebNLG (Table 3) look similar. Graformer outperforms most original challenge participants and more recent work. While not performing on par with CGE-LW on WebNLG, Graformer still achieves more than 96% of its performance while using only about half as many parameters.

5.2 Performance on different types of graphs

We investigate whether Graformer is more suitable for disconnected graphs by comparing its performance on different splits of the AGENDA test set according to two graph properties: (i) the average number of nodes per connected component (μ_c) and (ii) the largest diameter across all of a graph's components (d).

μ_c		BLEU	METEOR	CHRF++
< 1.25	Ours	**15.44**	20.59	43.23
(213)	CGE-LW	15.34	**20.64**	**43.56**
< 1.5	Ours	**17.45**	22.03	45.67
(338)	CGE-LW	17.29	**22.32**	**45.88**
< 2.0	Ours	18.94	22.86	46.49
(294)	CGE-LW	**19.46**	**23.76**	**47.78**
≥ 2.0	Ours	**21.72**	24.22	48.79
(155)	CGE-LW	20.97	**24.98**	**49.83**

(a) Average size μ_c of graph components.

d		BLEU	METEOR	CHRF++
1	Ours	**16.48**	**21.16**	43.94
(368)	CGE-LW	16.33	**21.16**	**44.16**
2	Ours	**18.46**	22.70	46.85
(414)	CGE-LW	18.20	**23.14**	**47.28**
≥ 3	Ours	19.44	23.17	47.29
(218)	CGE-LW	**20.32**	**24.42**	**49.25**

(b) Largest diameter d across all of a graph's components.

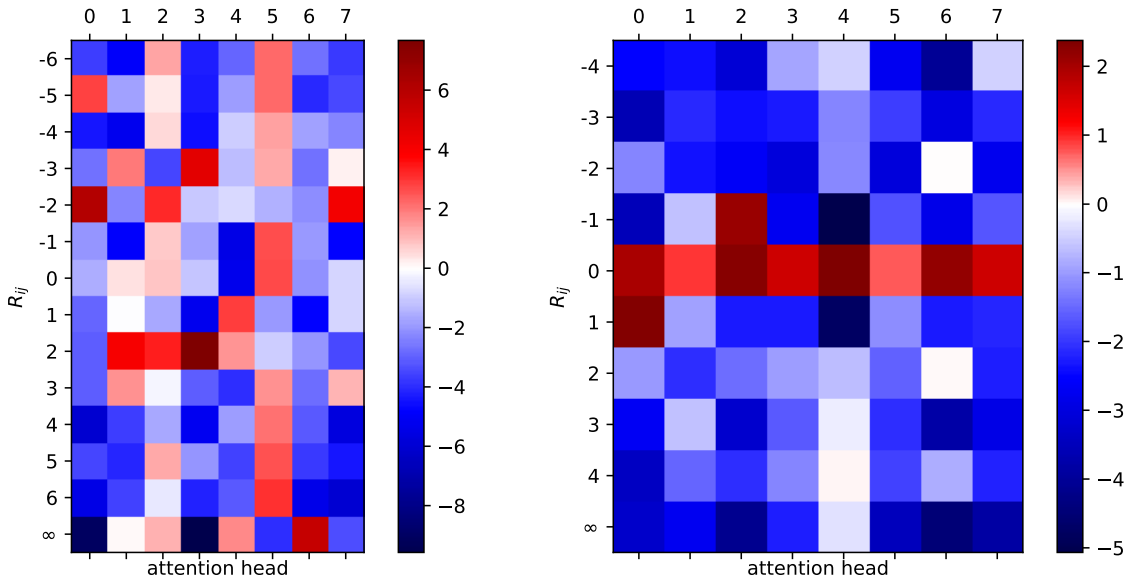
Table 4: Performance of a single run on the test split of AGENDA w.r.t. different input graph properties. The number of data points in each split is indicated in parentheses.

We can see in Table 4 that the performance of both Graformer and CGE-LW (Ribeiro et al., 2020) increases with more graph structure (larger μ_c and d), i.e., more information leads to more accurate texts. Besides, Graformer outperforms CGE-LW on BLEU for graphs with smaller components ($0 < \mu_c < 1.5$) and smaller diameters ($\mathsf{d} < 3$). Although METEOR and CHRF++ scores always favor CGE-LW, the performance difference is also smaller for cases where BLEU favors Graformer.

We conjecture that Graformer benefits from its more elaborate global view, i.e., its ability to distinguish between distant but connected nodes and unreachable ones. CGE-LW's global encoder cannot make this distinction because it only sees a fully connected version of the graph.

Curiously, Graformer's BLEU is also better for larger components ($\mu_c \geq 2.0$). With multiple larger components, Graformer might also better distinguish nodes that are part of the same component from those that belong to a different one.

Only for $1.5 < \mu_c < 2.0$, CGE-LW clearly outperforms Graformer in all metrics. It seems that Graformer is most helpful for extreme cases, i.e., when either most components are isolated nodes or when isolated nodes are the exception.



(a) AGENDA (b) WebNLG

Figure 3: Attention bias γ learned by Graformer on the two datasets. SAME_p edges are omitted.

Model	BLEU	METEOR	CHRF++
Graformer	18.09	22.29	45.77
-length penalty	17.99	22.19	45.63
-beam search	17.33	21.74	44.87
-epoch curriculum	13.55	18.91	39.22

Table 5: Ablation study for a single run on the test portion of AGENDA.

5.3 Ablation study

In a small ablation study, we examine the impact of beam search, length penalty, and our new epoch curriculum training. We find that beam search and length penalty do contribute to the overall performance but to a relatively small extent. Training with our new epoch curriculum, however, proves crucial for good performance. Platanios et al. (2019) argue that curriculum learning can replace a learning rate schedule, which is usually essential to train a Transformer model. Indeed we successfully optimize Graformer without any learning rate schedule, when applying the epoch curriculum.

6 Learned graph structure

We visualize the learned attention bias γ for different relative graph positions R_{ij} (cf. § 3.3; esp. Eq. (7)) after training on AGENDA and WebNLG in Fig. 3. The eight attention heads (x-axis) have learned different weights for each graph position R_{ij} (y-axis). Note that AGENDA has more possible R_{ij} values because $n_\delta = 6$ whereas we set $n_\delta = 4$ for WebNLG.

For both datasets, we notice that one attention head primarily focuses on global information (5 for AGENDA, 4 for WebNLG). AGENDA even dedicates head 6 entirely to unreachable nodes, showing the importance of such nodes for this dataset. In contrast, most WebNLG heads suppress information from unreachable nodes.

For both datasets, we also observe that nearer nodes generally receive a high weight (focus on local information): In Fig. 3b, e.g., head 2 concentrates solely on direct incoming edges and head 0 on direct outgoing ones. Graformer can learn empirically based on its task where direct neighbors are most important and where they are not, showing that the strong bias from graph neural networks is not necessary to learn about graph structure.

7 Conclusion

We presented Graformer, a novel encoder-decoder architecture for graph-to-text generation based on Transformer. The Graformer encoder uses a novel type of self-attention for graphs based on shortest path lengths between nodes, allowing it to detect global patterns by automatically learning appropriate weights for higher-order neighborhoods. In our experiments on two popular benchmarks for text generation from knowledge graphs, Graformer achieved competitive results while using many fewer parameters than alternative models.

Acknowledgments

This work was supported by the BMBF (first author) as part of the project MLWin (01IS18050), by the German Research Foundation (second author) as part of the Research Training Group "Adaptive Preparation of Information from Heterogeneous Sources" (AIPHES) under the grant No. GRK 1994/1, and by the Bavarian research institute for digital transformation (bidt) through their fellowship program (third author). We also gratefully acknowledge a Ph.D. scholarship awarded to the first author by the German Academic Scholarship Foundation (Studienstiftung des deutschen Volkes).

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Waleed Ammar, Dirk Groeneveld, Chandra Bhagavatula, Iz Beltagy, Miles Crawford, Doug Downey, Jason Dunkelberger, Ahmed Elgohary, Sergey Feldman, Vu Ha, Rodney Kinney, Sebastian Kohlmeier, Kyle Lo, Tyler Murray, Hsu-Han Ooi, Matthew Peters, Joanna Power, Sam Skjonsberg, Lucy Wang, Chris Wilhelm, Zheng Yuan, Madeleine van Zuylen, and Oren Etzioni. 2018. Construction of the literature graph in semantic scholar. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pages 84–91, New Orleans - Louisiana. Association for Computational Linguistics.

Bang An, Xuannan Dong, and Changyou Chen. 2019. Repulsive bayesian sampling for diversified attention modeling. *4th workshop on Bayesian Deep Learning (NeurIPS 2019)*.

Sören Auer, Chris Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2008. DBpedia: A nucleus for a web of open data. In *Proceedings of the 6th International Semantic Web Conference (ISWC)*, volume 4825 of *Lecture Notes in Computer Science*, pages 722–735. Springer.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

James S. Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2546–2554. Curran Associates, Inc.

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

Deng Cai and Wai Lam. 2020. Graph transformer for graph-to-sequence learning. *AAAI Conference on Artificial Intelligence*.

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

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. 2018. The best of both worlds: Combining recent advances in neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–86, Melbourne, Australia. Association for Computational Linguistics.

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

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

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *Computing Research Repository*, arXiv:1606.08415.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9:1735–80.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text Generation from Knowledge Graphs with Graph Transformers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2284–2293, Minneapolis, Minnesota. Association for Computational Linguistics.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Fei-Fei Li. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123:32–73.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Qimai Li, Zhichao Han, and Xiao ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. *AAAI Conference on Artificial Intelligence*.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3219–3232, Brussels, Belgium. Association for Computational Linguistics.

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

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019. Step-by-step: Separating planning from realization in neural data-to-text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2267–2277, Minneapolis, Minnesota. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Emmanouil Antonios Platanios, Otilia Stretcu, Graham Neubig, Barnabas Poczos, and Tom Mitchell. 2019. Competence-based curriculum learning for neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1162–1172, Minneapolis, Minnesota. Association for Computational Linguistics.

Maja Popović. 2017. chrF++: words helping character n-grams. In *Proceedings of the Second Conference on Machine Translation*, pages 612–618, Copenhagen, Denmark. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *Computing Research Repository*, arXiv:1910.10683.

Leonardo F. R. Ribeiro, Claire Gardent, and Iryna Gurevych. 2019. Enhancing AMR-to-text generation with dual graph representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3183–3194, Hong Kong, China. Association for Computational Linguistics.

Leonardo F. R. Ribeiro, Yue Zhang, Claire Gardent, and Iryna Gurevych. 2020. Modeling global and local node contexts for text generation from knowledge graphs. *Transactions of the Association for Computational Linguistics*, 8(0):589–604.

Martin Schmitt, Sahand Sharifzadeh, Volker Tresp, and Hinrich Schütze. 2020. An unsupervised joint system for text generation from knowledge graphs and semantic parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7117–7130, Online. Association for Computational Linguistics.

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

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4596–4604. PMLR.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, page 5998–6008. Curran Associates, Inc.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations (ICLR)*.

David S Wishart, Yannick D Feunang, An C Guo, Elvis J Lo, Ana Marcu, Jason R Grant, Tanvir Sajed, Daniel Johnson, Carin Li, Zinat Sayeeda, Nazanin Assempour, Ithayavani Iynkkaran, Yifeng Liu, Adam Maciejewski, Nicola Gale, Alex Wilson, Lucy Chin, Ryan Cummings, Diana Le, Allison Pon, Craig Knox, and Michael Wilson. 2018. DrugBank 5.0: a major update to the DrugBank database for 2018. *Nucleic Acids Research*, 46(D1):D1074–D1082.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *Computing Research Repository*, arXiv:1609.08144.

Kai Zhang, Yaokang Zhu, Jun Wang, and Jie Zhang. 2020. Adaptive structural fingerprints for graph attention networks. In *International Conference on Learning Representations (ICLR)*.

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

A Hyperparameter details

For AGENDA and WebNLG, a minimum and maximum decoding length were set according to the shortest and longest target text in the train set. Table 6 lists the hyperparameters used to obtain final results on both datasets. Input dropout is applied on the word embeddings directly after lookup for node labels and target text tokens before they are fed into encoder or decoder. Attention dropout is applied to all attention weights computed during multi-head (self-)attention.

Hyperparameter	WebNLG	AGENDA
model dimension d	256	400
# heads	8	8
# encoder layers L_E	3	4
# decoder layers L_D	3	5
feedforward dimension	512	2000
attention dropout	0.3	0.1
dropout	0.1	0.1
input dropout	0.0	0.1
text self-attention range n_t	25	50
graph self-attention range n_δ	4	6
SAME range n_p	10	10
gradient accumulation	3	2
gradient clipping	1.0	1.0
label smoothing	0.25	0.3
L_2 regularizer	$3 \cdot 10^{-3}$	$3 \cdot 10^{-4}$
batch size	4	8
# beams	2	2
length penalty	5.0	5.0

Table 6: Hyperparameters used to obtain final experimental results on WebNLG and AGENDA.

For hyperparameter optimization, we only train for the first 10 (AGENDA) or 50 (WebNLG) epochs to save time. We use a combination of manual tuning and a limited number of randomly sampled runs. For the latter we apply Optuna with default parameters (Akiba et al., 2019; Bergstra et al., 2011) and median pruning, i.e., after each epoch we check if the best performance so far is worse than the median performance of previous runs at the same epoch and if so, abort. For hyperparameter tuning, we decode greedily and measure performance in corpus-level BLEU (Papineni et al., 2002).

B Qualitative examples

Table 7 shows three example generations from our Graformer model and the CGE-LW system by Ribeiro et al. (2020). Often CGE-LW generations have a high surface overlap with the reference text while Graformer texts fluently express the same content.

Ref.	julia morgan has designed many significant buildings , including the los angeles herald examiner building .
CGE-LW	julia morgan has designed many significant buildings including the los angeles herald examiner building .
Ours	one of the significant buildings designed by julia morgan is the los angeles herald examiner building .
Ref.	asam pedas is a dish of fish cooked in a sour and hot sauce that comes from indonesia .
CGE-LW	the main ingredients of asam pedas are fish cooked in a sour and hot sauce and comes from indonesia .
Ours	the main ingredients of asam pedas are fish cooked in sour and hot sauce . the dish comes from indonesia .
Ref.	banana is an ingredient in binignit which is a dessert . a cookie is also a dessert .
CGE-LW	banana is an ingredient in binignit , a cookie is also a dessert .
Ours	a cookie is a dessert , as is binignit , which contains banana as one of its ingredients .

Table 7: Example references and texts generated by CGE-LW (Ribeiro et al., 2020) and Graformer (marked Ours) for samples from the WebNLG test set. In case of multiple references, only one is shown for brevity.

Chapter 7

Investigating Pretrained Language Models for Graph-to-Text Generation

Investigating Pretrained Language Models for Graph-to-Text Generation

Leonardo F. R. Ribeiro[†], **Martin Schmitt**[‡], **Hinrich Schütze**[‡] **and Iryna Gurevych**[†]

[†]Research Training Group AIPHES and UKP Lab, Technical University of Darmstadt
[‡]Center for Information and Language Processing (CIS), LMU Munich

www.ukp.tu-darmstadt.de

Abstract

Graph-to-text generation aims to generate fluent texts from graph-based data. In this paper, we investigate two recent pretrained language models (PLMs) and analyze the impact of different task-adaptive pretraining strategies for PLMs in graph-to-text generation. We present a study across three graph domains: meaning representations, Wikipedia knowledge graphs (KGs) and scientific KGs. We show that approaches based on PLMs BART and T5 achieve new state-of-the-art results and that task-adaptive pretraining strategies improve their performance even further. We report new state-of-the-art BLEU scores of 49.72 on AMR-LDC2017T10, 59.70 on WebNLG, and 25.66 on AGENDA datasets - a relative improvement of 31.8%, 4.5%, and 42.4%, respectively, with our models generating significantly more fluent texts than human references. In an extensive analysis, we identify possible reasons for the PLMs' success on graph-to-text tasks. Our findings suggest that the PLMs benefit from similar facts seen during pretraining or fine-tuning, such that they perform well even when the input graph is reduced to a simple bag of node and edge labels.[1]

1 Introduction

Graphs are important data structures in NLP as they represent complex relations within a set of objects. For example, semantic and syntactic structures of sentences can be represented using different graph representations (e.g., AMRs, Banarescu et al., 2013; semantic-role labeling, Surdeanu et al., 2008; syntactic and semantic graphs, Belz et al., 2011) and knowledge graphs (KGs) are used to describe factual knowledge in the form of relations between entities (Gardent et al., 2017; Vougiouklis et al., 2018; Koncel-Kedziorski et al., 2019).

Graph-to-text generation, a subtask of data-to-text generation (Gatt and Krahmer, 2018), aims to create fluent natural language text to describe an input graph (see Figure 1). This task is important for NLP applications such as dialogue generation (Moon et al., 2019) and question answering (Duan et al., 2017). Recently, it has been shown that structured meaning representation, such as AMR or KG, can store the internal state of a dialog system, providing core semantic knowledge (Bonial et al., 2020; Bai et al., 2021) or can be the result of a database query for conversational QA (Yu et al., 2019). Moreover, dialog states can be represented as KGs to encode compositionality and can be shared across different domains, slot types and dialog participators (Cheng et al., 2020).

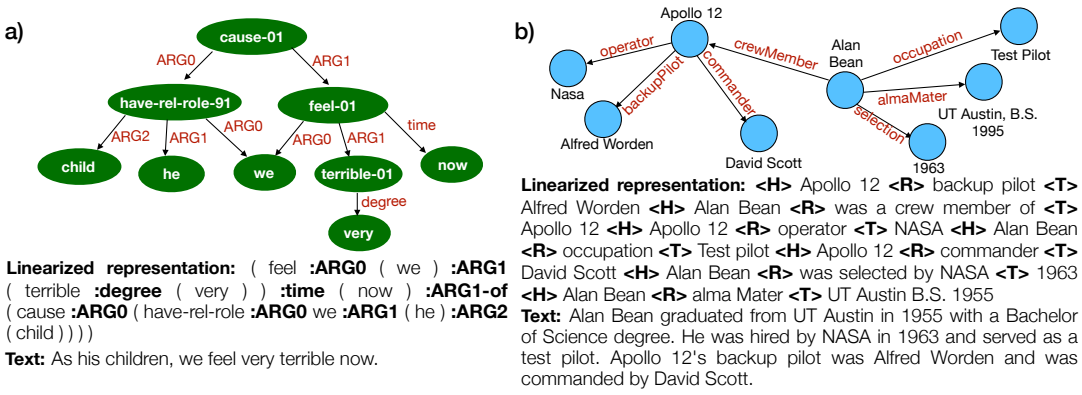
Transfer learning has become ubiquitous in NLP and pretrained Transformer-based architectures (Vaswani et al., 2017) have considerably outperformed prior state of the art in various downstream tasks (Devlin et al., 2019; Yang et al., 2019a; Liu et al., 2020; Radford et al., 2019).

In this paper, we analyze the applicability of two recent text-to-text pretrained language models (PLMs), BART (Lewis et al., 2020) and T5 (Raffel et al., 2019), for graph-to-text generation. We choose these models because of their *encoder-decoder* architecture, which makes them particularly suitable for conditional text generation. Our study comprises three graph domains (meaning representations, Wikipedia KGs, and scientific KGs). We further introduce *task-adaptive* graph-to-text pretraining approaches for PLMs and demonstrate that such strategies improve the state of the art by a substantial margin.

While recent works have shown the benefit of explicitly encoding the graph structure in graph-to-text generation (Song et al., 2018; Ribeiro et al., 2019, 2020; Schmitt et al., 2020; Zhao et al., 2020a, to name a few), our approaches based on PLMs consistently outperform these models, even though PLMs – as sequence models – do not exhibit any

[1]Our code is available at https://github.com/UKPLab/plms-graph2text.

Proceedings of the Third Workshop on Natural Language Processing for Conversational AI, pages 211–227
November 10, 2021.



Linearized representation: (feel **:ARG0** (we) **:ARG1** (terrible **:degree** (very)) **:time** (now) **:ARG1-of** (cause **:ARG0** (have-rel-role **:ARG0** we **:ARG1** (he) **:ARG2** (child))))
Text: As his children, we feel very terrible now.

Linearized representation: **<H>** Apollo 12 **<R>** backup pilot **<T>** Alfred Worden **<H>** Alan Bean **<R>** was a crew member of **<T>** Apollo 12 **<H>** Apollo 12 **<R>** operator **<T>** NASA **<H>** Alan Bean **<R>** occupation **<T>** Test pilot **<H>** Apollo 12 **<R>** commander **<T>** David Scott **<H>** Alan Bean **<R>** was selected by NASA **<T>** 1963 **<H>** Alan Bean **<R>** alma Mater **<T>** UT Austin B.S. 1955
Text: Alan Bean graduated from UT Austin in 1955 with a Bachelor of Science degree. He was hired by NASA in 1963 and served as a test pilot. Apollo 12's backup pilot was Alfred Worden and was commanded by David Scott.

Figure 1: Examples of (a) AMR and (b) WebNLG graphs, the input for the models and the reference texts.

graph-specific structural bias.[2] Simply representing the graph as a linear traversal (see Figure 1) leads to remarkable generation performance in the presence of a strong language model. In our analysis we investigate to what extent fine-tuned PLMs make use of the graph structure represented in the graph linearization. We notably observe that PLMs achieve high performance on two popular KG-to-text benchmarks even when the KG is reduced to a mere bag of node and edge labels.

Our contributions are the following:

- We investigate and compare two PLMs, BART and T5, for graph-to-text generation, exploring *language model adaptation* (LMA) and *supervised task adaptation* (STA) pretraining, employing additional task-specific data.
- Our approaches consistently outperform the state of the art by significant margins, ranging from 2.6 to 12.0 BLEU points, on three established graph-to-text benchmarks from different domains, exceeding specialized graph architectures (e.g., Graph Neural Networks, GNNs, Kipf and Welling, 2017).
- In a crowdsourcing experiment, we demonstrate that our methods generate texts with significantly better fluency than existing works and the human references.
- We discover that PLMs perform well even when trained on a shuffled linearized graph representation without any information about connectivity (bag of node and edge labels), which is surprising since prior studies showed that explicitly encoding the graph structure improves models trained from scratch (e.g., Zhao et al., 2020a); and investigate the possible reasons for such a good performance.

[2]The model architecture does not explicitly encode the graph structure, i.e., which entities are connected to each other, but has to retrieve it from a sequence that tries to encode this information.

2 Related Work

Graph-to-text Learning. Various neural models have been proposed to generate sentences from graphs from different domains. Konstas et al. (2017) propose the first neural approach for AMR-to-text generation that uses a linearized input graph. Prior approaches for KG-to-text generation train text-to-text neural models using sequences of KG triples as input (Trisedya et al., 2018; Moryossef et al., 2019; Castro Ferreira et al., 2019; Ribeiro et al., 2021a).

Recent approaches (Marcheggiani and Perez Beltrachini, 2018; Song et al., 2018; Beck et al., 2018; Damonte and Cohen, 2019; Ribeiro et al., 2019; Zhao et al., 2020a; Schmitt et al., 2021; Ribeiro et al., 2021b) propose architectures based on GNNs to directly encode the graph structure, whereas other efforts (Ribeiro et al., 2020; Schmitt et al., 2020; Yao et al., 2020; Wang et al., 2020) inject the graph structure information into Transformer-based architectures. The success of those approaches suggests that imposing a strong relational inductive bias into the graph-to-text model can assist the generation.

Pretrained Language Models. Pretrained Transformer-based models, such as BERT (Devlin et al., 2019), XLNet (Yang et al., 2019b), or RoBERTa (Liu et al., 2020), have established a qualitatively new level of baseline performance for many widely used natural language understanding (NLU) benchmarks. Generative pretrained Transformer-based methods, such as GPT-2 (Radford et al., 2019), BART (Lewis et al., 2020), and T5 (Raffel et al., 2019), are employed in many

natural language generation (NLG) tasks.

Mager et al. (2020) were the first to employ GPT-2, a decoder-only PLM, for AMR-to-text generation and use cycle consistency to improve the adequacy. In contrast, we are the first to investigate BART and T5 models, which have both a Transformer-based encoder and decoder, in AMR-to-text generation. Recently, Harkous et al. (2020) and Kale (2020) demonstrate state-of-the-art results in different data-to-text datasets, employing GPT-2 and T5 models respectively. Radev et al. (2020) propose DART, a new data-to-text dataset, and train a BART model gradually augmenting the WebNLG training data with DART data.

Hoyle et al. (2021) explore scaffolding objectives in PLMs and show gains in low-resource graph-to-text settings. Different from the above works, we focus on a general transfer learning strategies for graph-to-text generation, investigating task-adaptive pretraining approaches, employing additional collected task-specific data for different PLMs (BART and T5) and benchmarks. In addition, we provide a detailed analysis aimed at explaining the good performance of PLMs on KG-to-text tasks.

Recently, Gururangan et al. (2020) explored task-adaptive pretraining strategies for text classification. While our LMA (see §3) is related to their DAPT as both use a self-supervised objective on a domain-specific corpus, they notably differ in that DAPT operates on the model input while LMA models the output. We are the first to show the benefits of additional task-specific pretraining in PLMs for graph-to-text tasks.

3 PLMs for Graph-to-Text Generation

3.1 Models in this Study

We investigate BART (Lewis et al., 2020) and T5 (Raffel et al., 2019), two PLMs based on the Transformer *encoder-decoder* architecture (Vaswani et al., 2017), for graph-to-text generation. They mainly differ in how they are pretrained and the input corpora used for pretraining. We experiment with different T5 (*small* - 60M parameters, *base* - 220M, and *large* - 770M) and BART (*base* - 140M and *large* - 400M) capacity models.

We fine-tune both PLMs for a few epochs on the supervised downstream graph-to-text datasets. For T5, in the supervised setup, we add a prefix "translate from Graph to Text:" before the graph input. We add this prefix to imitate the T5 setup, when translating between different languages.

3.2 Task-specific Adaptation

Inspired by previous work (Konstas et al., 2017; Gururangan et al., 2020), we investigate whether leveraging additional task-specific data can improve the PLMs' performance on graph-to-text generation. Task-specific data refers to a pretraining corpus that is more task-relevant and usually smaller than the text corpora used for task-independent pretraining. In order to leverage the task-specific data, we add an intermediate adaptive pretraining step between the original pretraining and fine-tuning phases for graph-to-text generation.

More precisely, we first continue pretraining BART and T5 using language model adaptation (LMA) or supervised task adaptation (STA) training. In the supervised approach, we use pairs of graphs and corresponding texts collected from the same or similar domain as the target task. In the LMA approach, we follow BART and T5 pretraining strategies for language modeling, using the reference texts that describe the graphs. Note that we do not use the graphs in the LMA pretraining, but only the target text of our task-specific data collections. The goal is to adapt the decoder to the domain of the final task (Gururangan et al., 2020). In particular, we randomly mask text spans, replacing 15% of the tokens.[3] Before evaluation, we finally fine-tune the models using the original training set as usual.

4 Datasets

We evaluate the text-to-text PLMs on three graph-to-text benchmarks: AMR (LDC2017T10), WebNLG (Gardent et al., 2017), and AGENDA (Koncel-Kedziorski et al., 2019). We chose those datasets because they comprise different domains and are widely used in prior work. Table 10 in Appendix shows statistics for each dataset.

AMR. Abstract meaning representation (AMR) is a semantic formalism that represents the meaning of a sentence as a rooted directed graph expressing "who is doing what to whom" (Banarescu et al., 2013). In an AMR graph, nodes represent concepts and edges represent semantic relations. An instance in LDC2017T10 consists of a sentence annotated with its corresponding AMR graph. Following Mager et al. (2020), we linearize the AMR graphs

[3] Please, refer to Lewis et al. (2020) and Raffel et al. (2019) for details about the self-supervised pretraining strategies.

using the PENMAN notation (see Figure 1a).[4]

WebNLG. Each instance of WebNLG contains a KG from DBPedia (Auer et al., 2007) and a target text with one or multiple sentences that describe the graph. The test set is divided into two partitions: *seen*, which contains only DBPedia categories present in the training set, and *unseen*, which covers categories never seen during training. Their union is called *all*. Following previous work (Harkous et al., 2020), we prepend $\langle H \rangle$, $\langle R \rangle$, and $\langle T \rangle$ tokens before the head entity, the relation and tail entity of a triple (see Figure 1b).

AGENDA. In this dataset, KGs are paired with scientific abstracts extracted from proceedings of AI conferences. Each sample contains the paper title, a KG, and the corresponding abstract. The KG contains entities corresponding to scientific terms and the edges represent relations between these entities. This dataset has loose alignments between the graph and the corresponding text as the graphs were automatically generated. The input for the models is a text containing the title, a sequence of all KG entities, and the triples. The target text is the paper abstract. We add special tokens into the triples in the same way as for WebNLG.

4.1 Additional Task-specific Data

In order to evaluate the proposed task-adaptive pretraining strategies for graph-to-text generation, we collect task-specific data for two graph domains: meaning representations (like AMR) and scientific data (like AGENDA). We did not attempt collecting additional data like WebNLG because the texts in this benchmark do not stem from a corpus but were specifically written by annotators.

AMR Silver Data. In order to generate additional data for AMR, we sample two sentence collections of size 200K and 2M from the Gigaword[5] corpus and use a state-of-the-art AMR parser (Cai and Lam, 2020a) to parse them into AMR graphs.[6] For supervised pretraining, we condition a model on the AMR silver graphs to generate the corresponding sentences before fine-tuning it on gold AMR graphs. For self-supervised pretraining, we only use the sentences.[7]

[4]Details of the preprocessing procedure of AMRs are provided in Appendix A.

[5]https://catalog.ldc.upenn.edu/LDC2003T05

[6]We filter out sentences that do not yield well-formed AMR graphs.

[7]Gigaword and AMR datasets share similar data sources.

Model	BLEU	M	BT
Ribeiro et al. (2019)	27.87	33.21	-
Zhu et al. (2019)	31.82	36.38	-
Zhao et al. (2020b)	32.46	36.78	-
Wang et al. (2020)	33.90	37.10	-
Yao et al. (2020)	34.10	38.10	-
based on PLMs			
Mager et al. (2020)	33.02	37.68	-
Harkous et al. (2020)	37.70	38.90	-
$\text{BART}_{\text{base}}$	36.71	38.64	52.47
$\text{BART}_{\text{large}}$	43.47	42.88	60.42
T5_{small}	38.45	40.86	57.95
T5_{base}	42.54	42.62	60.59
T5_{large}	**45.80**	**43.85**	**61.93**
with task-adaptive pretraining			
$\text{BART}_{\text{large}}$ + LMA	43.94	42.36	58.54
T5_{large} + LMA	46.06	44.05	62.59
$\text{BART}_{\text{large}}$ + STA (200K)	44.72	43.65	61.03
$\text{BART}_{\text{large}}$ + STA (2M)	47.51	44.70	62.27
T5_{large} + STA (200K)	48.02	44.85	63.86
T5_{large} + STA (2M)	***49.72***	***45.43***	***64.24***

Table 1: Results on AMR-to-text generation for the LDC2017T10 test set. M and BT stand for METEOR and BLEURT, respectively. **Bold** (***Italic***) indicates the best score without (with) task-adaptive pretraining.

Semantic Scholar AI Data. We collect titles and abstracts of around 190K scientific papers from the Semantic Scholar (Ammar et al., 2018) taken from the proceedings of 36 top Computer Science/AI conferences. We construct KGs from the paper abstracts employing DyGIE++ (Wadden et al., 2019), an information extraction system for scientific texts. Note that the AGENDA dataset was constructed using the older SciIE system (Luan et al., 2018), which also extracts KGs from AI scientific papers. A second difference is that in our new dataset, the domain is broader as we collected data from 36 conferences compared to 12 from AGENDA. Furthermore, to prevent data leakage, all AGENDA samples used for performance evaluation are removed from our dataset. We will call the new dataset KGAIA (KGs from AI Abstracts).[8] Table 11 in Appendix shows relevant dataset statistics.

5 Experiments

We modify the BART and T5 implementations released by Hugging Face (Wolf et al., 2019) in order to adapt them to graph-to-text generation. For the KG datasets, we add the $\langle H \rangle$, $\langle R \rangle$, and $\langle T \rangle$ tokens to the models' vocabulary. We add all edge labels seen in the training set to the vocabulary of the

[8]We will release the collected additional task-specific data.

	BLEU			METEOR			chrF++		
Model	**A**	**S**	**U**	**A**	**S**	**U**	**A**	**S**	**U**
Castro Ferreira et al. (2019)	51.68	56.35	38.92	32.00	41.00	21.00	-	-	-
Moryossef et al. (2019)	47.24	53.30	34.41	39.00	44.00	37.00	-	-	-
Schmitt et al. (2020)	-	59.39	-	-	42.83	-	-	74.68	-
Ribeiro et al. (2020)	-	63.69	-	-	44.47	-	-	76.66	-
Zhao et al. (2020a)	52.78	64.42	38.23	41.00	46.00	37.00	-	-	-
based on PLMs									
Harkous et al. (2020)	52.90	-	-	42.40	-	-	-	-	-
Kale (2020)	57.10	63.90	52.80	44.00	46.00	41.00	-	-	-
Radev et al. (2020)	45.89	52.86	37.85	40.00	42.00	37.00	-	-	-
$\text{BART}_{\text{base}}$	53.11	62.74	41.53	40.18	44.45	35.36	70.02	76.68	62.76
$\text{BART}_{\text{large}}$	54.72	63.45	43.97	42.23	45.49	38.61	72.29	77.57	66.53
T5_{small}	56.34	**65.05**	45.37	42.78	45.94	39.29	73.31	**78.46**	67.69
T5_{base}	59.17	64.64	52.55	43.19	**46.02**	41.49	74.82	78.40	70.92
T5_{large}	**59.70**	64.71	**53.67**	**44.18**	45.85	**42.26**	**75.40**	78.29	**72.25**

Table 2: Results on WebNLG. A, S and U stand for *all*, *seen*, and *unseen* partitions of the test set, respectively.

models for AMR. Following Wolf et al. (2019), we use the Adam optimizer (Kingma and Ba, 2015) with an initial learning rate of $3 \cdot 10^{-5}$. We employ a linearly decreasing learning rate schedule without warm-up. The batch and beam search sizes are chosen from {2,4,8} and {1,3,5}, respectively, based on the respective development set. Dev BLEU is used for model selection.

Following previous works, we evaluate the results with BLEU (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2014), and chrF++ (Popović, 2015) metrics. We also use MoverScore (Zhao et al., 2019), BERTScore (Zhang et al., 2020), and BLEURT (Sellam et al., 2020) metrics, as they employ contextual and semantic knowledge and thus depend less on the surface symbols. Additionally, we perform a human evaluation (cf. §5.4) quantifying the fluency, semantic adequacy and meaning similarity of the generated texts.

5.1 Results on AMR-to-Text

Table 1 shows our results for the setting without additional pretraining, with additional self-supervised task-adaptive pretraining solely using the collected Gigaword sentences (LMA), and with additional supervised task adaptation (STA), before fine-tuning. We also report several recent results on the AMR test set. Mager et al. (2020) and Harkous et al. (2020) employ GPT-2 in their approaches. Note that GPT-2 only consists of a Transformer-based decoder.

Only considering approaches without task adaptation, $\text{BART}_{\text{large}}$ already achieves a considerable improvement of 5.77 BLEU and 3.98 METEOR scores over the previous state of the art. With a BLEU score of 45.80, T5_{large} performs best. The other metrics follow similar trends. See Table 13 in Appendix for evaluation with more metrics. The strong performance of both BART and T5 in the AMR dataset suggests that PLMs can infer the AMR structure by a simple linear sequence of the graph, in contrast to GNN-based models that explicitly consider the graph structure using *message-passing* between adjacent nodes (Beck et al., 2018).

Task-specific Adaptation. LMA already brings some gains with T5 benefitting more than BART in most metrics. It still helps less than STA even though we only have automatically generated annotations. This suggests that the performance increases with STA do not only come from additional exposure to task-specific target texts and that the models learn how to handle graphs and the graph-text correspondence even with automatically generated AMRs. After STA, T5 achieves 49.72 BLEU points, the new state of the art for AMR-to-text generation. Interestingly, gains from STA with 2M over 200K are larger in BART than in T5, suggesting that large amounts of silver data may not be required for a good performance with T5.

In general, models pretrained on the STA setup converge faster than without task-specific adaptation. For example, T5_{large} without additional pretraining converges after 5 epochs of fine-tuning whereas T5_{large} with STA already converges after 2 epochs.

5.2 Results on WebNLG

Table 2 shows the results for the WebNLG test set. Neural pipeline models (Moryossef et al., 2019; Castro Ferreira et al., 2019) achieve strong performance in the *unseen* dataset. On the other

Model	BLEU	M	BT
Koncel et al. 2019	14.30	18.80	-
An (2019)	15.10	19.50	-
Schmitt et al. (2020)	17.33	21.43	-
Ribeiro et al. (2020)	18.01	22.23	-
$\text{BART}_{\text{base}}$	22.01	23.54	-13.02
$\text{BART}_{\text{large}}$	**23.65**	**25.19**	**-10.93**
T5_{small}	20.22	21.62	-24.10
T5_{base}	20.73	21.88	-21.03
T5_{large}	22.15	23.73	-13.96
with task-adaptive pretraining			
$\text{BART}_{\text{large}}$ + LMA	25.30	25.54	-08.79
T5_{large} + LMA	22.92	24.40	-10.39
$\text{BART}_{\text{large}}$ + STA	***25.66***	***25.74***	***-08.97***
T5_{large} + STA	23.69	24.92	-08.94

Table 3: Results on AGENDA test set. **Bold** (***Italic***) indicates best scores without (with) task-adaptive pretraining.

Model	AMR	
	F	**MS**
Mager et al. (2020)	5.69^{A}	5.08^{A}
Harkous et al. (2020)	5.78^{A}	5.47^{AB}
T5_{large}	6.55^{B}	6.44^{C}
$\text{BART}_{\text{large}}$	6.70^{B}	5.72^{BC}
Reference	5.91^{A}	-
Model	**WebNLG**	
	F	**SA**
Castro Ferreira et al. (2019)	5.52^{A}	4.77^{A}
Harkous et al. (2020)	5.74^{AB}	6.21^{B}
T5_{large}	6.71^{C}	6.63^{B}
$\text{BART}_{\text{large}}$	6.53^{C}	6.50^{B}
Reference	5.89^{B}	6.47^{B}

Table 4: Fluency (F), Meaning Similarity (MS) and Semantic Adequacy (SA) obtained in the human evaluation. Differences between models which have a letter in common are not statistically significant and were determined by pairwise Mann-Whitney tests with $p < 0.05$.

hand, fully end-to-end models (Ribeiro et al., 2020; Schmitt et al., 2020) have strong performance on the *seen* dataset and usually perform poorly in *unseen* data. Models that *explicitly encode the graph structure* (Ribeiro et al., 2020; Zhao et al., 2020a) achieve the best performance among approaches that do not employ PLMs. Note that T5 is also used in Kale (2020). Differences in our T5 setup include a modified model vocabulary, the use of beam search, the learning rate schedule and the prefix before the input graph. Our T5 approach achieves 59.70, 65.05 and 54.69 BLEU points on *all*, *seen* and *unseen* sets, the new state of the art.

We conjecture that the performance gap between *seen* and *unseen* sets stems from the advantage obtained by a model seeing examples of relation-text pairs during fine-tuning. For example, the relation *party* (political party) was never seen during training and the model is required to generate a text that verbalizes the tuple: ⟨*Abdul Taib Mahmud, party, Parti Bumiputera Sarawak*⟩. Interestingly, BART performs much worse than T5 on this benchmark, especially in the *unseen* partition with 9.7 BLEU points lower compared to T5.

For lack of a suitable data source (cf. §4), we did not explore our LMA or STA approaches for WebNLG. However, we additionally discuss cross-domain STA in Appendix B.

5.3 Results on AGENDA

Table 3 lists the results for the AGENDA test set. The models also show strong performance on this dataset. We believe that their capacity to generate fluent text helps when generating paper abstracts, even though they were not pretrained in the scientific domain. $\text{BART}_{\text{large}}$ shows an impressive performance with a BLEU score of 23.65, which is 5.6 points higher than the previous state of the art.

Task-specific Adaptation. On AGENDA, BART benefits more from our task-adaptive pretraining, achieving the new state of the art of 25.66 BLEU points, a further gain of 2 BLEU points compared to its performance without task adaptation. The improvements from task-adaptive pretraining are not as large as for AMR. We hypothesize that this is due to the fact that the graphs do not completely cover the target text (Koncel-Kedziorski et al., 2019), making this dataset more challenging. See Table 12 in Appendix for more automatic metrics.

5.4 Human Evaluation

To further assess the quality of the generated text, we conduct a human evaluation on AMR and WebNLG via crowd sourcing on Amazon Mechanical Turk.[9] Following previous works (Gardent et al., 2017; Castro Ferreira et al., 2019), we assess three quality criteria: (i) *Fluency* (i.e., does the text flow in a natural, easy-to-read manner?), for AMR and WebNLG; (ii) *Meaning Similarity* (i.e., how

[9]We exclude AGENDA because its texts are scientific in nature and annotators are not necessarily AI experts.

Original Input
• Arrabbiata sauce • country • Italy • Italy • demonym • Italians • Italy • capital • Rome • Italy • language • Italian language • Italy • leader Name • Sergio Mattarella

Shuffle

Corrupted Input
• Rome • Italy • Italy • language • capital • Italy • Italians • Italy • Italy • Sergio Mattarella • Arrabbiata sauce • leader Name • country • demonym • Italian language

T5^{order}

Arrabbiata sauce can be found in Italy where Sergio Mattarella is the leader and the capital city is Rome. Italians are the people who live there and the language spoken is Italian.

T5^{shuf}

Italians live in Italy where the capital is Rome and the language is Italian. Sergio Mattarella is the leader of the country and arrabbiata sauce can be found there.

Reference: Arrabbiata sauce is from Italy where the capital is Rome, Italian is the language spoken and Sergio Mattarella is a leader.

Figure 2: Example graph with 5 triples, from WebNLG dev linearized with the neutral separator tag, denoted •, (top left), its shuffled version (top right), texts generated with two fine-tuned versions of T5$_{small}$ and a gold reference (bottom). Note that T5 can produce a reasonable text even when the input triples are shuffled randomly.

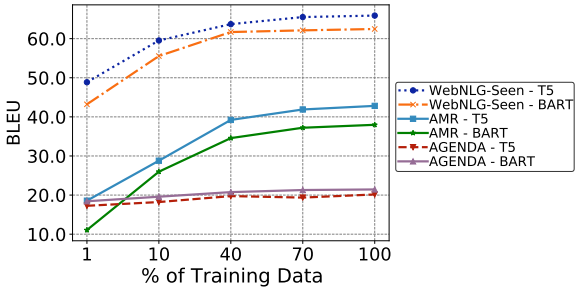


Figure 3: Performance of BART$_{base}$ and T5$_{base}$ in the dev set when experimenting with different amounts of training data.

close in meaning is the generated text to the reference sentence?) for AMR; (ii) *Semantic Adequacy* (i.e., does the text clearly express the data?) for WebNLG. We randomly select 100 generated texts of each model, which the annotators then rate on a 1-7 Likert scale. For each text, we collect scores from 3 annotators and average them.[10]

Table 4 shows the results. Our approaches improve the fluency, meaning similarity, and semantic adequacy on both datasets compared to other state-of-the-art approaches with statistically significant margins (p<0.05). Interestingly, the highest fluency improvement (+0.97) is on AMR, where our approach also has the largest BLEU improvement (+8.10) over Harkous et al. (2020). Finally, our models score higher than the references in fluency with statistically significant margins, highlighting their strong language generation abilities.[11]

5.5 Limiting the Training Data

In Figure 3, we investigate the PLMs' performance, measured with BLEU score, while varying (from 1% to 100%) the amount of training data used for fine-tuning. We find that, when fine-tuned with only 40% of the data, both BART and T5 already greatly improve the performance compared to using the entire training data in all three benchmarks. For example, BART fine-tuned on 40% of AMR training data achieves 91% of the BLEU score when fine-tuned on full data.

Note that in a low-resource scenario in AMR and WebNLG, T5 considerably outperforms BART. In particular, with only 1% of training examples, the difference between T5 and BART is 7.51 and 5.64 BLEU points for AMR and WebNLG, respectively. This suggests that T5 is more data efficient when adapting to the new task, likewise our findings in AMR-STA (cf. §5.1).

Model	AMR	WebNLG	AGENDA
T5^{order}	36.83	63.41	19.86
T5^{shuf}	15.56	61.54	19.08

Table 5: Impact (measured with BLEU) of using a bag of entities and relations (*shuf*) as input for T5$_{small}$.

6 Influence of the Graph Structure

We conduct further experiments to examine how much the PLMs consider the graph structure. To this end, we remove parentheses in AMRs and replace $\langle H \rangle$, $\langle R \rangle$, and $\langle T \rangle$ tokens with neutral separator tokens, denoted •, for KGs, such that the graph structure is only defined by the order of node and edge labels. If we shuffle such a sequence, the graph structure is thus completely obscured and the input effectively becomes a bag of node and edge labels. See Figure 2 for an example of both a correctly ordered and a shuffled triple sequence.

6.1 Quantitative Analysis

Table 5 shows the effect on T5's performance when its input contains correctly ordered triples (T5^{order})

[10] Inter-annotator agreement for the three criteria ranged from 0.40 to 0.79, with an average Krippendorff's α of 0.56.

[11] Examples of fluent generations can be found in the Tables 15 and 16 in Appendix.

	T/F	Input Fact	T5^{order}	T5^{shuf}
(1)	S	• German language • Antwerp • Antwerp • Antwerp International Airport • Belgium • Belgium • Charles Michel • city Served • leader Name • Belgium • language • country	Antwerp International Airport serves the city of Antwerp. German is the language spoken in Belgium where Charles Michel is the leader.	Antwerp International Airport serves the city of Antwerp in Belgium where the German language is spoken and Charles Michel is the leader.
(2)	T	• California • is Part Of • US • California • capital • Sacramento	California is part of the United States and its capital is Sacramento.	California is part of the United States and its capital is Sacramento.
(3)	F	• US • is Part Of • California • California • capital • Sacramento	California's capital is Sacramento and the United States is part of California.	California is part of the United States and its capital is Sacramento.
(4)	T	• Amarillo, Texas • is Part Of • United States	Amarillo, Texas is part of the United States.	Amarillo, Texas is part of the United States.
(5)	F	• United States • is Part Of • Amarillo, Texas	Amarillo, Texas is part of the United States.	Amarillo, Texas is part of the United States.

Table 6: Example generations from shuffled (S), true (T), and corrupted (F) triple facts by $T5_{small}$, fine-tuned on correctly ordered triples (*order*) and randomly shuffled input (*shuf*).

vs. shuffled ones (T5^{shuf}) for both training and evaluation. We first observe that T5^{order} only has marginally lower performance (around 2-4%) with the neutral separators than with the $\langle H\rangle/\langle R\rangle/\langle T\rangle$ tags or parentheses.[12] We see that as evidence that the graph structure is similarly well captured by T5^{order}. Without the graph structure (T5^{shuf}), AMR-to-text performance drops significantly. Possible explanations of this drop are: (i) the relative ordering of the AMR graph is known to correlate with the target sentence order (Konstas et al., 2017); (ii) in contrast to WebNLG that contains common knowledge, the AMR dataset contains very specific sentences with higher surprisal;[13] (iii) AMRs are much more complex graph structures than the KGs from WebNLG and AGENDA.[14]

On the other hand, KG-to-text performance is not much lower, indicating that most of the PLMs' success in this task stems from their language modeling rather than their graph encoding capabilities. We hypothesize that a PLM can match the entities in a shuffled input with sentences mentioning these entities from the pretraining or fine-tuning phase. It has recently been argued that large PLMs can recall certain common knowledge facts from pretraining (Petroni et al., 2019; Bosselut et al., 2019).

6.2 Qualitative Analysis

The example in Figure 2 confirms our impression. T5^{shuf} produces a text with the same content as T5^{order} but does not need the correct triple structure to do so. Example (1) in Table 6 shows the output of both models with shuffled input. Interestingly, even T5^{order} produces a reasonable and truthful text. This suggests that previously seen facts serve as a strong guide during text generation, even for models that were fine-tuned with a clearly marked graph structure, suggesting that T5^{order} also relies more on language modeling than the graph structure. It does have more difficulties covering the whole input graph though. The fact that *Antwerp* is located in *Belgium* is missing from its output.

To further test our hypothesis that PLMs make use of previously seen facts during KG-to-text generation, we generate example true facts, corrupt them in a controlled setting, and feed them to both T5^{order} and T5^{shuf} to observe their output (examples (2)–(5) in Table 6). The model trained on correctly ordered input has learned a bit more to rely on the input graph structure. The false fact in example (3) with two triples is reliably transferred to the text by T5^{order} but not by T5^{shuf}, which silently corrects it. Also note that, in example (5), both models refuse to generate an incorrect fact. More examples can be found in Table 14 in the Appendix.

Our qualitative analysis illustrates that state-of-the-art PLMs, despite their fluency capacities (cf. §5.4), bear the risk of parroting back training sentences while ignoring the input structure. This issue can limit the practical usage of those models as, in many cases, it is important for a generation model to stay true to its input (Wiseman et al., 2017; Falke et al., 2019).

[12] See a more fine-grained comparison in Appendix C.

[13] Perplexities estimated on the dev sets of AMR and WebNLG datasets, with GPT-2 fine-tuned on the corresponding training set, are 20.9 and 7.8, respectively.

[14] In Appendix D, we present the graph properties of the datasets and discuss the differences.

7 Conclusion

We investigated two pretrained language models (PLMs) for graph-to-text generation and show that the pretraining strategies, language model adaptation (LMA) and supervised task adaptation (STA), can lead to notable improvements. Our approaches outperform the state of the art by a substantial margin on three graph-to-text benchmarks. Moreover, in a human evaluation our generated texts are perceived significantly more fluent than human references. Examining the influence of the graph structure on the text generation process, we find that PLMs may not always follow the graph structure and instead use memorized facts to guide the generation. A promising direction for future work is to explore ways of injecting a stronger graph-structural bias into PLMs, thus possibly leveraging their strong language modeling capabilities and keeping the output faithful to the input graph.

Acknowledgments

We thank our anonymous reviewers for their thoughtful feedback. Leonardo F. R. Ribeiro is supported by the German Research Foundation (DFG) as part of the Research Training Group "Adaptive Preparation of Information form Heterogeneous Sources" (AIPHES, GRK 1994/1) and as part of the DFG funded project UKP-SQuARE with the number GU 798/29-1. Martin Schmitt is supported by the BMBF as part of the project MLWin (01IS18050) and by the German Academic Scholarship Foundation (Studienstiftung des deutschen Volkes).

References

Waleed Ammar, Dirk Groeneveld, Chandra Bhagavatula, Iz Beltagy, Miles Crawford, Doug Downey, Jason Dunkelberger, Ahmed Elgohary, Sergey Feldman, Vu Ha, Rodney Kinney, Sebastian Kohlmeier, Kyle Lo, Tyler Murray, Hsu-Han Ooi, Matthew Peters, Joanna Power, Sam Skjonsberg, Lucy Wang, Chris Wilhelm, Zheng Yuan, Madeleine van Zuylen, and Oren Etzioni. 2018. Construction of the literature graph in semantic scholar. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pages 84–91, New Orleans - Louisiana. Association for Computational Linguistics.

Bang An. 2019. Repulsive bayesian sampling for diversified attention modeling. In *4th workshop on Bayesian Deep Learning (NeurIPS 2019)*.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *Proceedings of the 6th International The Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference*, ISWC'07/ASWC'07, page 722–735, Berlin, Heidelberg. Springer-Verlag.

Xuefeng Bai, Yulong Chen, Linfeng Song, and Yue Zhang. 2021. Semantic representation for dialogue modeling. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4430–4445, Online. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

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

Anja Belz, Michael White, Dominic Espinosa, Eric Kow, Deirdre Hogan, and Amanda Stent. 2011. The first surface realisation shared task: Overview and evaluation results. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 217–226, Nancy, France. Association for Computational Linguistics.

Claire Bonial, Lucia Donatelli, Mitchell Abrams, Stephanie M. Lukin, Stephen Tratz, Matthew Marge, Ron Artstein, David Traum, and Clare Voss. 2020. Dialogue-AMR: Abstract Meaning Representation for dialogue. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 684–695, Marseille, France. European Language Resources Association.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. COMET: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779, Florence, Italy. Association for Computational Linguistics.

Deng Cai and Wai Lam. 2020a. AMR parsing via graph-sequence iterative inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1290–1301, Online. Association for Computational Linguistics.

Deng Cai and Wai Lam. 2020b. Graph transformer for graph-to-sequence learning. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7464–7471. AAAI Press.

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

Jianpeng Cheng, Devang Agrawal, Héctor Martínez Alonso, Shruti Bhargava, Joris Driesen, Federico Flego, Dain Kaplan, Dimitri Kartsaklis, Lin Li, Dhivya Piraviperumal, Jason D. Williams, Hong Yu, Diarmuid Ó Séaghdha, and Anders Johannsen. 2020. Conversational semantic parsing for dialog state tracking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8107–8117, Online. Association for Computational Linguistics.

Marco Damonte and Shay B. Cohen. 2019. Structural neural encoders for AMR-to-text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3649–3658, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 866–874, Copenhagen, Denmark. Association for Computational Linguistics.

Tobias Falke, Leonardo F. R. Ribeiro, Prasetya Ajie Utama, Ido Dagan, and Iryna Gurevych. 2019. Ranking generated summaries by correctness: An interesting but challenging application for natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2214–2220, Florence, Italy. Association for Computational Linguistics.

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

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61(1):65–170.

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

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

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

Alexander Miserlis Hoyle, Ana Marasović, and Noah A. Smith. 2021. Promoting graph awareness in linearized graph-to-text generation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 944–956, Online. Association for Computational Linguistics.

Mihir Kale. 2020. Text-to-text pre-training for data-to-text tasks. *arXiv e-prints*.

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

Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR 2017.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text Generation from Knowledge Graphs with Graph Transformers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2284–2293, Minneapolis, Minnesota. Association for Computational Linguistics.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural amr: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157, Vancouver, Canada. Association for Computational Linguistics.

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

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Roberta: A robustly optimized bert pretraining approach. *arXiv e-prints*.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3219–3232, Brussels, Belgium. Association for Computational Linguistics.

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

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

Seungwhan Moon, Pararth Shah, Anuj Kumar, and Rajen Subba. 2019. OpenDialKG: Explainable conversational reasoning with attention-based walks over knowledge graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 845–854, Florence, Italy. Association for Computational Linguistics.

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019. Step-by-step: Separating planning from realization in neural data-to-text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2267–2277, Minneapolis, Minnesota. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

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

Dragomir Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Nazneen Fatema Rajani, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, Yangxiaokang Liu, Nadia Irwanto, Jessica Pan, Faiaz Rahman, Ahmad Zaidi, Murori Mutuma, Yasin Tarabar, Ankit Gupta, Tao Yu, Yi Chern Tan, Xi Victoria Lin, Caiming Xiong, and Richard Socher. 2020. Dart: Open-domain structured data record to text generation. *arXiv e-prints*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *arXiv e-prints*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*.

Leonardo F. R. Ribeiro, Claire Gardent, and Iryna Gurevych. 2019. Enhancing AMR-to-text generation with dual graph representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3183–3194, Hong Kong, China. Association for Computational Linguistics.

Leonardo F. R. Ribeiro, Jonas Pfeiffer, Yue Zhang, and Iryna Gurevych. 2021a. Smelting gold and silver for improved multilingual amr-to-text generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Punta Cana, November 7-11, 2021*.

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

Leonardo F. R. Ribeiro, Yue Zhang, and Iryna Gurevych. 2021b. Structural adapters in pretrained language models for amr-to-text generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Punta Cana, November 7-11, 2021*.

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

Martin Schmitt, Leonardo F. R. Ribeiro, Philipp Dufter, Iryna Gurevych, and Hinrich Schütze. 2020. Modeling graph structure via relative position for better text generation from knowledge graphs. *arXiv e-prints*.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

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

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL 2008 shared task on joint parsing of syntactic and semantic dependencies. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177, Manchester, England. Coling 2008 Organizing Committee.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Pavlos Vougiouklis, Hady Elsahar, Lucie-Aimée Kaffee, Christophe Gravier, Frédérique Laforest, Jonathon Hare, and Elena Simperl. 2018. Neural wikipedian: Generating textual summaries from knowledge base triples. *Journal of Web Semantics*, 52-53:1 – 15.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5784–5789, Hong Kong, China. Association for Computational Linguistics.

Tianming Wang, Xiaojun Wan, and Hanqi Jin. 2020. Amr-to-text generation with graph transformer. *Transactions of the Association for Computational Linguistics*, 8:19–33.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

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

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019a. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32, pages 5753–5763. Curran Associates, Inc.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019b. Xlnet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 5753–5763. Curran Associates, Inc.

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

Linguistics, pages 7145–7154, Online. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter Lasecki, and Dragomir Radev. 2019. CoSQL: A conversational text-to-SQL challenge towards cross-domain natural language interfaces to databases. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1962–1979, Hong Kong, China. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Chao Zhao, Marilyn Walker, and Snigdha Chaturvedi. 2020a. Bridging the structural gap between encoding and decoding for data-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2481–2491, Online. Association for Computational Linguistics.

Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M. Meyer, and Steffen Eger. 2019. MoverScore: Text generation evaluating with contextualized embeddings and earth mover distance. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 563–578, Hong Kong, China. Association for Computational Linguistics.

Yanbin Zhao, Lu Chen, Zhi Chen, Ruisheng Cao, Su Zhu, and Kai Yu. 2020b. Line graph enhanced AMR-to-text generation with mix-order graph attention networks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 732–741, Online. Association for Computational Linguistics.

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

Appendices

In this supplementary material, we provide: (i) additional information about the data used in the experiments, and (ii) results that we could not fit into the main body of the paper.

A AMR Input Representation

We test three variants for the representation of the input AMR graph. Following previous work (Konstas et al., 2017; Mager et al., 2020), we evaluate (i) only node representation, where the edge information is removed from the linearization; (ii) depth-first search (DFS) through the graph and the (iii) PENMAN representation. An example for each representation is illustrated below:

only nodes
```
value interrogative commodity
true
```

DFS
```
value :mode interrogative
:ARG1 commodity :ARG1-of
true
```

PENMAN
```
( value :mode interrogative
:ARG1 ( commodity ) :ARG1-of
( true ) )
```

In this experiment we employ $T5_{small}$. Table 7 shows the results on the AMR development set. The PENMAN representation leads to best results. Therefore, this representation is used in the rest of the experiments.

Input	BLEU
only nodes	28.22
DFS	34.94
PENMAN	38.27

Table 7: Results on the AMR dev set using $T5_{small}$ for different AMR linearizations.

B Cross-domain Adaptation

For a given task, it is not always possible to collect closely related data – as we saw, e.g., for WebNLG. We therefore report STA in a cross-domain setting for the different KG-to-text benchmarks. Table 8 shows the results using $BART_{base}$ and $T5_{base}$. While the texts in KGAIA and AGENDA share the domain of scientific abstracts, texts in WebNLG are more general. Also note that WebNLG graphs do not share any relations with the other KGs. For $BART_{base}$, STA increases the performance in the cross-domain setting in most of the cases. For $T5_{base}$, STA in KGAIA improves the performance on WebNLG.

In general, we find that exploring additional adaptive pretraining for graph-to-text generation can improve the performance even if the data do not come from the same domain.

STA on	Fine-tuned & Evaluated on	
	WebNLG-*Seen*	AGENDA
	$BART_{base}$	
None	58.71	22.01
KGAIA	63.20	23.48
WebNLG	-	21.98
AGENDA	61.25	-
	$T5_{base}$	
None	62.93	20.73
KGAIA	63.19	22.44
WebNLG	-	20.27
AGENDA	62.75	-

Table 8: Effect (measured with BLEU score) of cross-domain STA.

C Input Graph Size

Figure 4 visualizes $T5_{small}$'s performance with respect to the number of input graph triples in WebNLG dataset. We observe that $T5^{order}$ and $T5^{shuf}$ perform similarly for inputs with only one triple but that the gap between the models increases with larger graphs. While it is obviously more difficult to reconstruct a larger graph than a smaller one, this also suggests that the graph structure is more taken into account for graphs with more than 2 triples. For the *unseen* setting, the performance gap for these graphs is even larger, suggesting that the PLM can make more use of the graph structure when it has to.

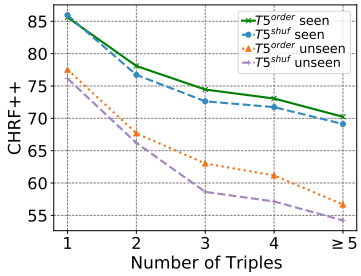


Figure 4: chrF++ scores with respect to the number of triples for WebNLG *seen* and *unseen* test sets.

D Graph Statistics

In Table 9, we present the graph properties of the three datasets. All statistics are calculated using

	AMR			WebNLG			AGENDA		
min, avg and max number of nodes	2	28.6	335	2	6.8	15	2	10.5	80
min, avg and max node degrees	1	2.2	21	1	1.7	7	1	1.67	15
min, avg and max number of edges	1	32.3	554	1	5.9	14	1	8.8	124
min, avg and max graph diameter	1	12.2	40	1	4.1	10	1	3.1	20
min, avg and max shortest path length	0	7.49	40	0	2.4	10	0	2.3	20

Table 9: Graph statistics of AMR, WebNLG and AGENDA datasets. The values are calculated using the training data. Note that AMR graphs contain a more complex structure than WebNLG and AGENDA graphs.

the Levi transformation (Beck et al., 2018) of the undirected version of the graphs, where edges are also considered nodes in the graph. WebNLG and AGENDA datasets contain disconnected graphs, and we use the largest subgraph to calculate the diameter. Note that AMR graphs have a much more complex structure: (i) they have more nodes and edges than WebNLG and AGENDA graphs; (ii) the average graph diameter and the average shortest path between nodes in AMRs are at least three times larger than in WebNLG and AGENDA graphs; (iii) nodes in AMRs have larger degrees than nodes in WebNLG and AGENDA graphs.

	AMR17	WebNLG	AGENDA
#Train	36,521	18,102	38,720
#Dev	1,368	872	1,000
#Test	1,371	1,862	1,000
#Relations	155	373	7
Avg #Tokens	16.1	31.5	157.9

Table 10: Statistics for the graph-to-text benchmarks.

	Title	Abstract	KG
Vocab	48K	173K	113K
Tokens	2.1M	31.7M	9.6M
Entities	-	-	3.7M
Avg Length	11.1	167.1	-
Avg #Nodes	-	-	19.9
Avg #Edges	-	-	9.4

Table 11: Statistics for the KGAIA dataset.

Model	chrF++	BS (F1)	MS
Schmitt et al. (2020)	44.53	-	-
Ribeiro et al. (2020)	46.37	-	-
$BART_{base}$	48.02	89.36	34.33
$BART_{large}$	**50.44**	88.74	32.24
$T5_{small}$	44.91	88.56	30.25
$T5_{base}$	48.14	88.81	31.33
$T5_{large}$	48.14	**89.60**	**35.23**
with task-adaptive pretraining			
$BART_{large}$ + LMA	51.33	89.12	33.42
$T5_{large}$ + LMA	49.37	89.75	36.13
$BART_{large}$ + STA	***51.63***	89.27	34.28
$T5_{large}$ + STA	50.27	***89.93***	***36.86***

Table 12: Results of the chrF++, BertScore (BS) and MoverScore (MS) scores for AGENDA test set. **Bold** (***Italic***) indicates best scores without (with) task-adaptive pretraining.

Model	chrF++	BS (F1)	MS
Guo et al. (2019)	57.30	-	-
Zhu et al. (2019)	64.05	-	-
Cai and Lam (2020b)	59.40	-	-
Wang et al. (2020)	65.80	-	-
Yao et al. (2020)	65.60	-	-
based on PLMs			
Mager et al. (2020)	63.89	-	-
$BART_{base}$	66.65	95.22	60.78
$BART_{large}$	71.06	96.08	65.74
$T5_{small}$	68.78	95.62	63.70
$T5_{base}$	70.81	95.99	65.63
$T5_{large}$	**72.57**	**96.27**	**67.37**
with task-adaptive pretraining			
$BART_{large}$ + LMA	71.14	95.94	64.75
$T5_{large}$ + LMA	72.83	96.32	67.44
$BART_{large}$ + STA (200K)	72.26	96.21	66.75
$BART_{large}$ + STA (2M)	73.58	96.43	68.14
$T5_{large}$ + STA (200K)	74.09	96.51	68.86
$T5_{large}$ + STA (2M)	***74.79***	***96.59***	***69.53***

Table 13: Results of the chrF++, BertScore (BS) and MoverScore (MS) scores for the LDC2017T10 test set. **Bold** (***Italic***) indicates the best score without (with) task-adaptive pretraining.

	T/F	Input Facts	T5^{order}	T5^{shuf}
(1)	S	• capital • leader Name • London • Pound sterling • United Kingdom • leader Name • United Kingdom • Elizabeth II • United Kingdom • Boris Johnson • London • currency	The capital city is London, the currency is the Pound sterling and the leader is Elizabeth II. Boris Johnson is also a leader in the UK.	The capital of the United Kingdom is London, the currency is the Pound sterling and the country is lead by Elizabeth II and Boris Johnson.
(2)	T	• Germany • capital • Berlin	Berlin is the capital of Germany.	Berlin is the capital of Germany.
(3)	F	• Berlin • capital • Germany	Berlin's capital is Germany.	Berlin is the capital of Germany.
(4)	F	• Leinster • is Part Of • Dublin	Leinster is part of Dublin.	Leinster is part of Dublin.
(5)	F	• Rome • capital • Italy	Rome's capital is Italy.	Rome is the capital of Italy.
(6)	T	• Italy • capital • Rome	Italy's capital is Rome.	Rome is the capital of Italy.
(7)	T	• Texas • capital • Austin • Andrews County Airport • location • Texas	Austin is the capital of Texas where Andrews County Airport is located.	Austin is the capital of Texas where Andrews County Airport is located.
(8)	F	• Austin • capital • Texas • Andrews County Airport • location • Texas	The capital of Austin is Texas and Andrews County Airport is located in Texas.	Andrews County Airport is located in Texas where Austin is the capital.

Table 14: Example generations from shuffled (S), true (T), and corrupted (F) triple facts by $T5_{small}$, fine-tuned on correctly ordered triples (*order*) and randomly shuffled input (*shuf*).

D	Model	Examples
AMR	Reference	I had to deal with verbal abuse from my dad for a long 8 years before I came to uni and honestly, the only reason why I'm here is because it was the only way out.
	T5	I had to deal with 8 years of verbal abuse from my dad before coming to university and honestly the only reason I'm here is because it's the only way out.
	BART	I had to deal with my dad's verbal abuse for 8 years long before coming to uni and honestly the only reason I'm here is because it's the only way out.
	Mager et al. (2020)	i've had to deal with verbal abuse from my dad for 8 years (before i came to uni i was honestly the only reason i was here) and it's only because of the way it is.
WebNLG	Reference	Aaron Turner is an electric guitar player who has played with the black metal band Twilight and with Old Man Gloom. Death metal is a musical fusion of black metal.
	T5	Aaron Turner plays the electric guitar and is associated with the band Twilight. He is also a member of the Old Man Gloom band. Black metal and death metal are both genres of music.
	BART	The black metal genre is a variation of the death metal genre. It is also where the band, Twilight, and the alternative rock band, Old Man Gloom, are from. One of the members of the band is Aaron Turner, who plays the electric guitar.
	Harkous et al. (2020)	Aaron Turner, a.k.a. Black Metal, is a member of the Twilight (band) and Old Man Gloom bands. He also plays electric guitar and has a strong connection with the Death Metal genre.

Table 15: Examples of text generated by the different models. D refers to the dataset.

Model	Examples
Reference	in this paper, a new array signal processing technique by using particle swarm optimization is proposed to identify multipath channel parameters. the proposed array signal processing technique provides estimates to the channel parameters by finding a global minimum of an optimization problem. since the optimization problem is formulated in the cross-ambiguity function (caf) domain of the transmitted signal and the received array outputs, the proposed array signal processing technique is called as particle swarm optimization. the performance of the particle swarm optimization is compared with the space alternating generalized expectation maximization technique and with another recently proposed pso based technique for various snr values. simulation results indicate the superior performance of the pso based technique over mentioned techniques for all snr values.
T5	this paper, we propose an array signal processing technique based on particle swarm optimization to estimate the multipath channel parameters from the received and the transmitted signal in a cross-ambiguity domain. the proposed array signal processing technique is formulated as an optimization problem and particle swarm optimization is compared with the generalized expectation maximization technique. simulation results show that the proposed particle swarm optimization outperforms the pso based technique for a wide range of snr values. in addition, the proposed particle swarm optimization is computationally more efficient than the gem based technique for small snr values.
BART	in this paper, a new array signal processing technique based on particle swarm optimization is proposed. the proposed array signal processing technique is used to estimate the multipath channel parameters from the transmitted signal. the proposed array signal processing technique is formulated as an optimization problem in the cross-ambiguity domain. the particle swarm optimization is compared with the generalized expectation maximization technique. simulation results show that the proposed particle swarm optimization outperforms the pso based technique for all snr values. furthermore, the proposed particle swarm optimization is able to estimate the channel parameters more accurately than the generalized expectation maximization technique.
Ribeiro et al. (2020)	in this paper, a novel array signal processing technique based on particle swarm optimization is proposed to estimate the multipath channel parameters from the transmitted signal. the proposed array signal processing technique uses particle swarm optimization to estimate the multipath channel parameters. the proposed array signal processing technique is formulated as an optimization problem. simulation results show that the proposed array signal processing technique outperforms the conventional generalized expectation maximization technique and the pso based technique is robust to the snr values.

Table 16: Examples of text generated by the different models trained on the AGENDA dataset.

Chapter 8

Structural Adapters in Pretrained Language Models for AMR-to-Text Generation

Structural Adapters in Pretrained Language Models for AMR-to-Text Generation

Leonardo F. R. Ribeiro[†], **Yue Zhang**[‡] **and Iryna Gurevych**[†]

[†]Ubiquitous Knowledge Processing Lab, Technical University of Darmstadt
[‡]School of Engineering, Westlake University
ribeiro@aiphes.tu-darmstadt.de

Abstract

Pretrained language models (PLM) have recently advanced graph-to-text generation, where the input graph is linearized into a sequence and fed into the PLM to obtain its representation. However, efficiently encoding the graph structure in PLMs is challenging because such models were pretrained on natural language, and modeling structured data may lead to catastrophic forgetting of distributional knowledge. In this paper, we propose StructAdapt, an adapter method to encode graph structure into PLMs. Contrary to prior work, StructAdapt effectively models interactions among the nodes based on the graph connectivity, only training graph structure-aware adapter parameters. In this way, we incorporate task-specific knowledge while maintaining the topological structure of the graph. We empirically show the benefits of explicitly encoding graph structure into PLMs using StructAdapt, outperforming the state of the art on two AMR-to-text datasets, training only 5.1% of the PLM parameters.[1]

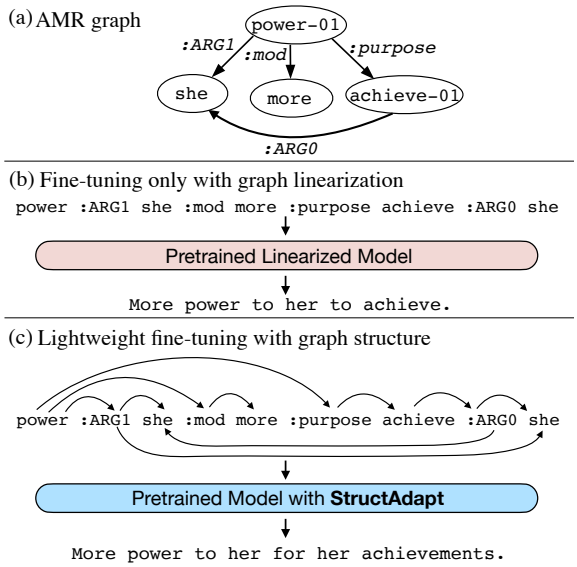


Figure 1: (a) AMR for the sentence *More power to her for her achievements*. While in (b) the pretrained model gets as input the graph linearization, in (c) it additionally receives the graph connectivity information.

1 Introduction

Data-to-text tasks aim to generate meaningful and coherent natural language text that faithfully conveys *structured data*. Some examples of structured information include tables (Parikh et al., 2020), Knowledge Graphs (KGs) (Gardent et al., 2017; Vougiouklis et al., 2018) and Abstract Meaning Representation (AMR) (Banarescu et al., 2013). In this work, we focus on AMR-to-text generation where the goal is to generate a fluent and grammatical sentence that is faithful to a given AMR graph (See Figure 1a). AMR is a semantic formalism that has received much research interest (Song et al., 2018; Guo et al., 2019; Ribeiro et al., 2019; Opitz et al., 2020, 2021; Fu et al., 2021) and has been shown to benefit downstream tasks such as text summarization (Liao et al., 2018) and machine translation (Song et al., 2019). Both statistical (Flanigan et al., 2016; Pourdamghani et al., 2016) and neural methods (Bai et al., 2020; Cai and Lam, 2020) have been investigated for AMR-to-text generation, and dominant methods make use of Graph Neural Networks (GNNs) (Kipf and Welling, 2017) or Transformers (Vaswani et al., 2017) for representing the input graph.

Pretrained language models (PLMs) (Devlin et al., 2019; Liu et al., 2020; Radford et al., 2019; Lewis et al., 2020) have been shown useful as a general text representation method, giving much improved results on a wide range of tasks (Wang et al., 2018, 2019). However, they cannot be directly leveraged to benefit AMR-to-text generation, and more generally graph-to-text generation, due to the structural nature of the input. One solution is to transform the structured input into a se-

[1]Our code and checkpoints are available at https://github.com/UKPLab/StructAdapt.

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 4269–4282
November 7–11, 2021.

quence, which can be directly fed into PLMs (See Figure 1b). Recent studies (Mager et al., 2020; Harkous et al., 2020; Ribeiro et al., 2020a, 2021) transform AMRs into sequences by top-down linearization (Konstas et al., 2017). It has been shown that such linearized graph representation can be used to fine-tune a PLM and improve graph-to-text generation performances (Kale, 2020).

The above methods, however, suffer from two salient limitations. First, linearized graph structures are different in nature from natural language. As a result, knowledge from large-scale pretraining intuitively cannot be fully transferred, and fine-tuning a sentence representation using linearized graphs can lead to catastrophic forgetting of such distributional knowledge (Goodfellow et al., 2014; Kirkpatrick et al., 2017). Second, a linearized representation weakens structural information in the original graphs by diluting the explicit connectivity information (i.e., which nodes are connected to each other), and PLMs must infer how edge connections are specified in the sequence. This fact was also observed by Song et al. (2018), Beck et al. (2018) and Ribeiro et al. (2019), who show that GNN encoders outperform sequential encoders for AMR-to-text generation without pretraining.

To mitigate the issues, we aim to explicitly encode the graph data into a PLM without contaminating its original distributional knowledge. To this end, we propose StructAdapt, a novel structure-aware adapter that effectively allows leveraging the input graph structure into PLMs (See Figure 1c). The main idea is to add layer-wise modules, which extract information from the pretrained layers and make use of it in a graph-structure encoding. As shown in Figure 2, StructAdapt employs a *graph convolution* in order to learn representations built upon the graph connectivity over the PLM encoder. Because StructAdapt is added to each encoder layer, deep integration of linguistic knowledge and graph knowledge can be achieved. During fine-tuning, only the adapter parameters are trained, whereas the PLM parameters remain unchanged, in contrast to previous methods based on the graph linearizations that fine-tune all model parameters.

Empirically we show that StructAdapt significantly outperforms linearized fine-tuning baselines and naive sequential adapters (Houlsby et al., 2019). Moreover, StructAdapt is more robust to different graph linearizations, better treats reentrancies (nodes with more than one entering edge) and long-range node dependencies. Our proposed models, based on StructAdapt, surpass the current state of the art on LDC2017T10 and LDC2020T02 datasets by up to 3.1 BLEU points, training only 5.1% of the original PLM parameters.

2 Related Work

Fine-tuning for Graph-to-text Generation. While previous approaches (Song et al., 2018; Ribeiro et al., 2019; Cai and Lam, 2020; Schmitt et al., 2021; Zhang et al., 2020b) have shown that explicitly encoding the graph structure is beneficial, fine-tuning PLMs on linearized structured data has established a new level of performance in data-to-text generation (Nan et al., 2021; Kale, 2020; Ribeiro et al., 2021). Our work can be seen as integrating the advantage of both graph structure encoding and PLMs, using a novel adapter module.

Mager et al. (2020) employ cycle consistency to improve the adequacy of generated texts from AMRs using GPT-2 (Radford et al., 2019), whereas Harkous et al. (2020) train a classifier to rank candidate generations based on the semantic fidelity. Ribeiro et al. (2020a) investigate encoder-decoder PLMs for graph-to-text generation, and show that task-specific pretraining can lead to notable improvements and that PLMs benefit much more from the graph structure of AMRs than of KGs. Hoyle et al. (2021) explore the extent to which PLMs are invariant to graph linearization, finding that models trained on canonical linearizations fail to generalize to meaning-preserving alternatives. Compared to this line of work, which tunes all PLM parameters, our method obtains a further 19x reduction in task-specific parameters, tuning only 5.1% of the parameters while achieving state-of-the-art performance, being more robust to permutations of the graph representation and better encoding larger graphs.

Lightweight Fine-tuning. Recently, different approaches have emerged as an alternative training strategy in order to avoid fine-tuning all parameters of a PLM. Zhang et al. (2019) train a lightweight "side" network that is fused with the pretrained model via summation. Li and Liang (2021) propose to prepend a trainable continuous prefix as an alternative to adapters, maintaining comparable performance in data-to-text tasks using fewer trained parameters. Liu et al. (2021) develop a method to automatically search prompts in the continuous space and evaluate it in few-shot NLU tasks. Ham-

bardzumyan et al. (2021) propose adversarial reprogramming attempts to learn task-specific word embeddings to customize the language model for the downstream task.

Adapter-based approaches (Houlsby et al., 2019; Rebuffi et al., 2017; Lauscher et al., 2020; Pfeiffer et al., 2020a, 2021) introduce a small number of task specific parameters, keeping the underlying pretrained model fixed. Pfeiffer et al. (2020b) propose an adapter method to arbitrary tasks and languages by learning modular language and task representations. The above works are related to StructAdapt as it trains much fewer parameters, but also different because they do not explicitly encode the input structure, whereas StructAdapt directly aims to encode it.

3 Graph-to-Text Model

Let $\mathcal{G}_0 = (\mathcal{V}_0, \mathcal{E}_0, \mathcal{R}_0)$ denote a rooted and directed AMR graph with a node set $\mathcal{V}_0$ and labeled edges $(u, r, v) \in \mathcal{E}_0$, where $u, v \in \mathcal{V}_0$ and $r \in \mathcal{R}_0$ is a relation type. An example of an AMR graph and its corresponding sentence is shown in Figure 1a.

3.1 Encoder-Decoder Architecture

Consider a conditional generation task where the input is a context x and the output $y = \langle y_1, \ldots, y_{|y|} \rangle$ is a sequence of tokens. In AMR-to-text generation, the context x is the AMR graph and y is the sentence that describes the AMR graph in natural language.

Let $p_\phi(y \mid x)$ denote a PLM parametrized by ϕ, where x is encoded by a bidirectional encoder, and the decoder predicts y autoregressively, conditioned on the encoded x and its left context. We focus on PLMs based on the Transformer encoder-decoder architecture (Vaswani et al., 2017), as they are suitable for conditional text generation. We define $x = \text{LIN}(\mathcal{G}_0)$, where LIN is a function that linearizes $\mathcal{G}_0$ into a sequence of tokens.[2] Following Damonte and Cohen (2019), as shown in Figure 1b, we linearize the AMR into a sequence of nodes and edges using the depth-first traversal of the canonical human-created AMR.[3] In a nutshell, the hidden representation $\mathbf{h}_i^l \in \mathbb{R}^d$, for all $x_i \in x$, is computed by the encoder layer l, where d is the hidden dimension. The decoder hidden representation $\hat{\mathbf{h}}_i^l \in \mathbb{R}^d$ is computed by the layer l of the autoregressive decoder at time step i.

[2]The variable of a re-entrant node – node with more than one incoming edge – is replaced with its co-referring concept.

[3]Other AMR linearizations are discussed in §6.1.

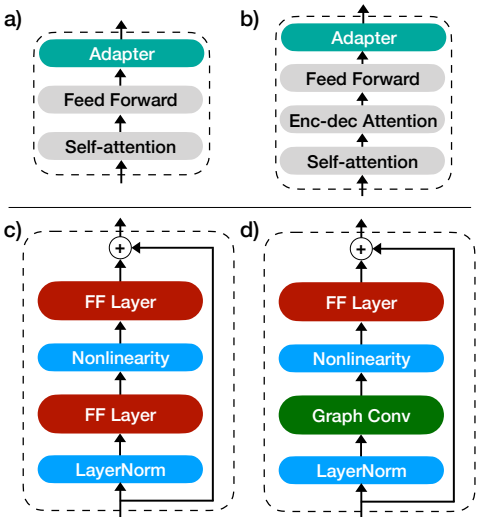


Figure 2: Integration of the adapter modules with the (a) encoder and (b) decoder layers of the Transformer; layer normalization and residual connections are omitted for clarification. (c) Adapt with two feed-forwards layers. (d) StructAdapt encodes the graph structure using a *graph convolutional layer*.

3.2 Fine-tuning

The model is initialized with pretrained parameters ϕ (e.g. using T5, Raffel et al., 2019) and fine-tuned to optimize the following log-likelihood objective over each gold instance (x, y):

$$\max_{\phi} \ \log p_\phi(y \mid x) = \sum_{i=1}^{|y|} \log p_\phi(y_i \mid y_{1:i-1}, x). \tag{1}$$

3.3 Baseline Adapter

We employ an adapter module after the feed-forward sub-layer of each layer on both encoder (Figure 2a) and decoder (Figure 2b) of the PLM. We modify the adapter architecture from Houlsby et al. (2019), computing the adapter representation at each layer l, given the encoder layer representation $\mathbf{h}_i^l$ (or $\hat{\mathbf{h}}_i^l$ in the decoder), as follows:

$$\hat{\mathbf{z}}_i = \mathbf{W}_o^l(\sigma(\mathbf{W}_p^l \, \text{LN}(\mathbf{h}_i^l))) + \mathbf{h}_i^l \,, \tag{2}$$

where σ is the activation function and $\text{LN}(\cdot)$ denotes layer normalization. $\mathbf{W}_o^l \in \mathbb{R}^{d \times m}$ and $\mathbf{W}_p^l \in \mathbb{R}^{m \times d}$ are adapter parameters, and m is the hidden dimension of the adapter. Figure 2c illustrates the baseline adapter module, which we call Adapt.

Training. Let the set of adapters' parameters for the encoder and decoder layers be parametrized by θ. The training objective is the same as Equation (1), but the set of trainable parameters changes: the PLM parameters ϕ are frozen and the adapter parameters θ are the only trainable parameters. In contrast to fine-tuning, adapters substantially reduce the number of trainable parameters that are used to adapt the PLM to the downstream task.

3.4 Limitation

Intuitively, the connection between nodes in the input graph can influence the encoding of x by guiding what to extract from x in order to generate y. Note that in both fine-tuning and ADAPT approaches, the self-attention mechanisms of the encoder layers treat the sequence of nodes and edges x essentially as a fully connected graph, greatly diluting the original graph structure. In this way, the model has to retrieve the original connectivity of the graph from x. For example, the AMR linearization in Figure 1b has two mentions of the node *she*, and the model should capture that both mentions belong to the same node in the original graph.

4 Structural Adapter

We propose STRUCTADAPT, a lightweight alternative to injecting *structural inductive bias*[4] into PLMs.

We first describe the intuition in §4.1 and define our method formally in §4.3.

4.1 Intuition

Injecting *graph structural bias* into graph-to-text models trained from scratch improves the performance compared to linearized approaches (Damonte and Cohen, 2019; Ribeiro et al., 2019). However, it is not straightforward how to effectively model the input graph structure when fine-tuning PLMs, which usually are pretrained using natural language and not structured data.

Our key idea is modeling the graph connectivity in the encoder utilizing an adapter module, using information flows between adjacent nodes in a message-passing update, employing a *graph convolution* (see Figure 2d). In this way, the graph structure substantially impacts the node representations, better encoding the input graph without impacting the knowledge learned during pretraining. This can lead to more efficient and better AMR-to-text generation as we will show in §5 and §6. Moreover, different adapters for distinct graph domains can be used with the same PLM, yielding a high degree of parameter sharing for graph-to-text tasks.

[4]The model architecture explicitly encodes the graph structure, i.e., which nodes are connected to each other.

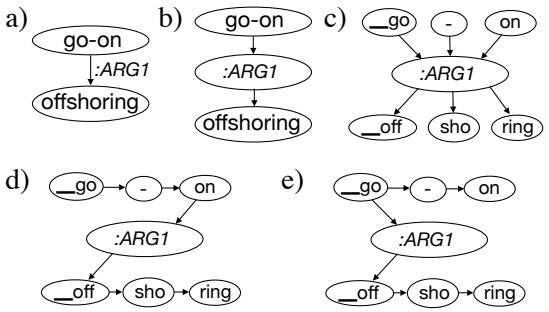


Figure 3: An example of (a) an AMR graph structure, (b) its unlabeled version and three different subword representations: (c) *rep1*, (d) *rep2* and (e) *rep3*.

4.2 Graph Representation

We convert each $\mathcal{G}_0$ into a bipartite graph $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$, replacing each labeled edge $(u, r, v) \in \mathcal{E}_0$ with two unlabeled edges $e_1 = (u, r)$ and $e_2 = (r, v)$. Similar to Beck et al. (2018), this process converts the graph into its unlabeled version. Figure 3 shows an (a) AMR subgraph and (b) its unlabeled representation.

Note that PLMs typically use a vocabulary with subword units (Sennrich et al., 2016). This presents a challenge in how to represent such a graph using subword tokens. Inspired by Ribeiro et al. (2020b), we transform each $\mathcal{G}_1$ into a new token graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each token of a node in $\mathcal{V}_1$ becomes a node $v \in \mathcal{V}$. We convert each edge $(u_1, v_1) \in \mathcal{E}_1$ into a set of edges and connect every token of u_1 to every token of v_1. That is, an edge (u, v) will belong to $\mathcal{E}$ if and only if there exists an edge $(u_1, v_1) \in \mathcal{E}_1$ such that $u \in u_1$ and $v \in v_1$, where u_1 and v_1 are seen as sets of tokens. Figure 3c shows an example of the token graph.

4.3 Method

STRUCTADAPT employs a two-layer architecture in order to re-purpose the PLM for the graph-to-text task using a small number of new parameters. Formally, for each node $v \in \mathcal{V}$, given the hidden representation $\mathbf{h}_v^l$ from the encoder layer l, STRUCTADAPT computes:

$$\mathbf{g}_v^l = \mathsf{GraphConv}_l(\mathsf{LN}(\mathbf{h}_v^l), \{\mathsf{LN}(\mathbf{h}_u^l) : u \in \mathcal{N}(v)\})$$
$$\mathbf{z}_v^l = \mathbf{W}_e^l \sigma(\mathbf{g}_v^l) + \mathbf{h}_v^l \,, \quad (3)$$

where $\mathcal{N}(v)$ is the immediate neighborhood of v in $\mathcal{G}$. $\mathsf{GraphConv}_l(\cdot)$ is the graph convolution that computes the node representation based on the *local neighborhood* of v, and $\mathbf{W}_e^l \in \mathbb{R}^{d \times m}$ is a parameter. Figure 2d illustrates StructAdapt.[5]

Graph Convolution. The graph convolutional layer allows exploration of distinct strategies for neighborhood aggregation in order to model structural information of the input graph. Different GNN architectures (Velickovic et al., 2018; Xu et al., 2019) can be employed as the graph convolution. Moreover, in this way, we avoid changing the self-attention mechanism of the current pretrained encoder, allowing to also capture *global information* based on the pretrained knowledge.

Our graph convolution is based on the Graph Convolutional Network (GCN) proposed by Kipf and Welling (2017). At each layer l, we compute the representation of a node $v \in \mathcal{V}$ as follows:

$$\mathbf{g}_v^l = \sum_{u \in \mathcal{N}(v)} \frac{1}{\sqrt{d_v d_u}} \mathbf{W}_g^l \mathbf{h}_u^l \ , \qquad (4)$$

where $\mathcal{N}(v)$ is a set of nodes with incoming edges to v and v itself, d_v is the degree of v, and $\mathbf{W}_g^l \in \mathbb{R}^{m \times d}$ is a parameter.

We also consider the variant relational GCN (RGCN) (Schlichtkrull et al., 2018) as graph convolution. RGCN allows capturing the reverse edge direction so that we can consider the differences in the incoming and outgoing relations, which has shown to be beneficial (Beck et al., 2018). In particular, the node representation is computed as:

$$\mathbf{g}_v^l = \sum_{r \in \mathcal{R}} \sum_{u \in \mathcal{N}_r(v)} \frac{1}{|\mathcal{N}_r(v)|} \mathbf{W}_r^l \mathbf{h}_u^l \ , \qquad (5)$$

where $\mathcal{R}$ denotes the set of relations, i.e., the edge types *default* and *reverse*, $\mathcal{N}_r(v)$ denotes the set of neighbors under relation $r \in \mathcal{R}$, and $\mathbf{W}_r^l \in \mathbb{R}^{m \times d}$ encodes the edge type between the nodes u and v.

Note that StructAdapt computes the refined structural node representation $\mathbf{z}_v^l$ based on the local node context, using as input the global representation $\mathbf{h}_v^l$ generated by the current PLM encoder layer. In this way, the model is able to capture both the global context based on the PLM linguistic knowledge and the local context based on the graph knowledge. Finally, we employ Adapt into the decoder in order to adapt the language model to the graph-to-text task.

[5]Preliminary experiments with other architecture configurations led to worse or similar performance.

	BLEU	chrF++	$\mathcal{M}$	BERT
Mager et al. (2020)	33.0	63.9	-	-
Zhang et al. (2020b)	33.6	63.2	-	-
Harkous et al. (2020)	37.7	-	-	-
Hoyle et al. (2021)	44.9	-	76.54	-
Ribeiro et al. (2020a)	45.8	72.5	-	-
$T5_{base}$				
Fine-tune	38.3±0.3	68.6±0.1	77.8±0.3	95.5±0.1
FT-top2(14.8%)	29.9±0.1	63.0±0.1	74.1±0.2	94.4±0.2
FT-bottom2(14.8%)	35.9±0.3	67.0±0.2	76.9±0.1	95.3±0.1
Adapt(8.5%)	38.7±0.4	69.2±0.2	78.3±0.1	95.6±0.1
StructAdapt-GCN(2.1%)	39.0±0.3	69.1±0.2	78.4±0.2	95.7±0.2
StructAdapt-GCN(8.5%)	41.0±0.5	70.0±0.2	78.4±0.1	95.7±0.1
StructAdapt-RGCN(6.3%)	**44.0**±0.3	**71.2**±0.2	**79.4**±0.1	**95.9**±0.2
$T5_{large}$				
Fine-tune	41.2±0.5	70.2±0.2	78.0±0.1	95.8±0.2
FT-top2(7.9%)	28.8±0.4	61.8±0.5	73.9±0.2	94.1±0.2
FT-bottom2(7.9%)	37.6±0.3	68.0±0.2	77.2±0.2	95.5±0.1
Adapt(6.8%)	42.9±0.3	71.6±0.2	78.9±0.1	96.1±0.1
StructAdapt-GCN(1.7%)	44.1±0.4	71.8±0.3	79.1±0.1	96.1±0.2
StructAdapt-GCN(6.8%)	45.8±0.2	72.5±0.1	79.3±0.2	96.2±0.1
StructAdapt-RGCN(5.1%)	**46.6**±0.3	**72.9**±0.2	**79.6**±0.1	**96.3**±0.1

Table 1: Results on the LDC2017T10 test set. Mean (±s.d.) over 4 seeds.

5 Experiments

Our models are initialized with pre-trained T5 (Raffel et al., 2019), but our approach can be combined with other PLMs such as BART (Lewis et al., 2020). Our implementation is based on Hugging Face Transformer models (Wolf et al., 2019). We use $T5_{base}$ for all experiments and report results with $T5_{large}$ for the test sets.[6] We use the Adam optimizer (Kingma and Ba, 2015) and employ a linearly decreasing learning rate schedule without warm-up. BLEU is used for the stopping criterion. Following recent work (Mager et al., 2020; Zhang et al., 2020b), we evaluate our proposed models on LDC2017T10 and LDC2020T02 corpora.

Evaluation. We evaluate the results with BLEU (Papineni et al., 2002) and chrF++ (Popović, 2015) metrics. We also report the meaning ($\mathcal{M}$) component of the $\mathcal{MF}$-score (Opitz and Frank, 2021), which measures how well the source AMR graph can be reconstructed from the generated sentence. We use BERTScore (Zhang et al., 2020a) allowing a semantic evaluation that depends less on the surface forms. Finally, we also perform a human evaluation (§5.2).

5.1 Main Results

We compare StructAdapt with four methods: fine-tuning (Fine-tune), fine-tuning only the top or bottom 2 layers (FT-top2, FT-bottom2) and Adapt. All

[6]Hyperparameter details are in the appendix A.

	BLEU	chrF++	$\mathcal{M}$	BERT
Zhang et al. (2020b)	34.3	63.7	-	-
Bevilacqua et al. (2021)	44.9	72.9	-	-
	$T5_{large}$			
FINE-TUNE	41.6±0.6	70.4±0.5	78.5±0.2	96.0±0.1
FT-TOP2(7.9%)	33.4±0.5	63.5±0.3	73.4±0.4	94.3±0.1
FT-BOTTOM2(7.9%)	38.2±0.2	68.3±0.1	78.1±0.2	95.6±0.1
ADAPT(6.8%)	43.0±0.2	71.3±0.2	79.3±0.1	96.2±0.1
STRUCTADAPT-GCN(1.7%)	46.2±0.2	71.8±0.2	79.4±0.3	96.0±0.2
STRUCTADAPT-GCN(6.8%)	47.1±0.4	72.5±0.1	79.7±0.2	96.2±0.1
STRUCTADAPT-RGCN(5.1%)	**48.0**±0.2	**73.2**±0.1	**80.1**±0.3	**96.3**±0.1

Table 2: Results on the LDC2020T02 test set.

Graph Size	ADAPT	STRUCTADAPT-RGCN
All	5.6^{A}	6.1^{B}
01-30	6.1^{A}	6.2^{A}
31-60	5.4^{A}	5.4^{A}
>60	5.2^{A}	6.2^{B}

Table 3: Meaning similarity obtained in the human evaluation. The ranking was determined by Mann-Whitney tests with $p<0.05$. Difference between systems which have a letter in common is not statistically significant.

models use the same graph linearization generated by the depth-first traversal. We also report recent state-of-the-art results on both datasets. Tables 1 and 2 show the results.

We find that training only 5.1% task-specific parameters, STRUCTADAPT-RGCN achieves a BLEU score of 46.6 in LDC2017T10, substantially improving over FINE-TUNE and other lightweight baselines (ADAPT, FT-TOP2, FT-BOTTOM2), and outperforming Ribeiro et al. (2020a) and Hoyle et al. (2021) which fine-tune T5 updating significantly more parameters. STRUCTADAPT also achieves state-of-the-art performance on LDC2020T02, considerably improving over Bevilacqua et al. (2021), which implicitly models the graph structure information using linearization techniques.

In general, STRUCTADAPT is better than ADAPT when training the same number of parameters, and slightly better even when training only 1.7% of the parameters for both datasets. This highlights that the gains not only come from using an adapter architecture, but from considering the graph connectivity. STRUCTADAPT-RGCN is more effective than STRUCTADAPT-GCN using fewer parameters, demonstrating that considering reverse relations is advantageous. ADAPT is consistently better than FINE-TUNE, agreeing with our intuition of catastrophic forgetting when fine-tuning. Interestingly, in contrast to popular strategies that focus on upper layers in fine-tuning (Howard and Ruder, 2018; Houlsby et al., 2019; Li and Liang, 2021), FT-BOTTOM2's performance is better than FT-TOP2's, suggesting that lower layers have a significant impact in adapting the PLM to structured data.

Different from our work, both Mager et al. (2020) and Ribeiro et al. (2020a) use the PENMAN notation which makes the input much longer (containing more tokens), and demonstrate that this representation is able to achieve strong results – this is orthogonal to our STRUCTADAPT representation and can be incorporated in future work.

Overall, the results indicate that explicitly considering the graph structure using an adapter mechanism is effective for AMR-to-text generation, significantly reducing the number of trained parameters while improving generation quality.

5.2 Human Evaluation

To further assess the quality of the generated texts by the adapter-based models in LDC2020T02, we conduct a human evaluation via crowdsourcing using Amazon Mechanical Turk. We follow previous work (Ribeiro et al., 2019; Castro Ferreira et al., 2019) and evaluate the *meaning similarity*, i.e., how close in meaning is the generated text to the reference sentence.[7] We divide the datapoints into 3 different sets by by the graph size, i.e., the number of nodes, after converting edges into nodes (cf. §4.2). This setting allows us to evaluate the performance of the models based on the complexity of the AMR graph.

We randomly select 100 generated texts for each set and each model (total of 600), which annotators then rate on a 1-7 Likert scale. For each text we collect scores from 3 annotators and use MACE (Hovy et al., 2013), a Bayesian model that incorporates the reliability of individual workers, to merge sentence-level labels.[8] Table 3 shows that STRUCTADAPT improves the meaning similarity over ADAPT with statistically significant margins ($p<0.05$). Note that the gains mainly come from datapoints with >60 nodes, indicating that STRUCTADAPT is better when encoding larger graphs.

5.3 Detailed Discussion

Parameter/Performance Trade-off. We investigate how the number of parameters affects the models. A higher hidden dimensionality means more

[7] We also assessed the *fluency* of the texts and the differences between the models were not statistically significant.

[8] Refer to Appendix B for a detailed description of the human evaluation.

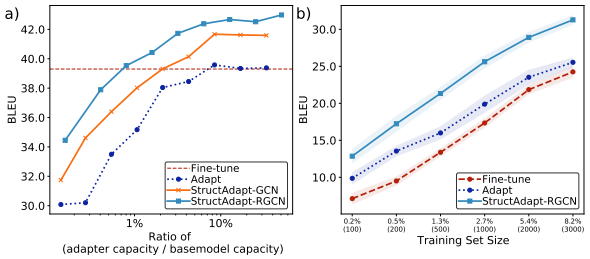


Figure 4: (a) Impact (measure with BLEU) of the number of parameters in the LDC2017T10 dev set. (b) Performance in the LDC2017T10 test set when experimenting with different amounts of training data.

trainable parameters, and smaller adapters introduce fewer parameters at a possible cost to performance. That is, the adapter size controls the parameter efficiency. Figure 4a shows the effect of the number of trained parameters in the performance measured using BLEU. Each point in the ADAPT and STRUCTADAPT curves represents a hidden dimension in the range $[8, 16, \ldots, 2048]$. STRUCTADAPT-GCN is consistently better than ADAPT over all model capacities, even though both approaches train the same number of parameters. STRUCTADAPT-RGCN achieves similar performance than FINE-TUNE when training only 0.8% of the parameters whereas ADAPT achieves similar performance to 8.5%, demonstrating the effectiveness of injecting the graph structure into the PLM.

Low-data Setting. Previous work (Li and Liang, 2021) has shown that lightweight fine-tuning has an advantage in some generation tasks when the training size is smaller. Therefore, we investigate how STRUCTADAPT behaves in a low-data setting. We subsample the LDC2017T10 training set to analyze different smaller training sets. For each size, we sample 5 different datasets and average over 2 training random seeds. Thus, we average over 10 models to get an estimate for each low-data setting.[9] Figure 4b shows the results. First note that both adapter-based approaches improve over FINE-TUNE. When training with only 1000 datapoints, STRUCTADAPT outperforms FINE-TUNE by 8.2 BLEU points. Also note that the gap between ADAPT and FINE-TUNE decreases when the size of the training set increases. In general, STRUCTADAPT outperforms FINE-TUNE and ADAPT in low-resource scenarios by 7.3 and 4.8 BLEU points on average, respectively, whereas requiring much fewer trained parameters than FINE-TUNE and fewer number of parameters than ADAPT.

[9]We use the LDC2017T10 dev set to choose hyperparameters and do early stopping.

(b / break-up-08 :ARG1 **(i / i)** :ARG3 (p / person :ARG0-of (h / have-rel-role-91 :ARG1 (p2 / person :ARG0-of (h2 / have-rel-role-91 :ARG1 **i** :ARG2 (s3 / son))) :ARG2 (f / father))) :time (s2 / since :op1 (d / date-entity :month 8)))
REFERENCE: Me and my son's father have been broken up since August.
FINE-TUNE-2000: I've broken up with my son and father since August.
FINE-TUNE: I've been with my son's father since August.
STRUCTADAPT-2000: Since August 8 I have broken up with my son's father.
STRUCTADAPT: I've been breaking up with my son's father since August.

Table 4: An example of an AMR graph and generated sentences by different models trained on full data and on a low-data setting with 2000 datapoints.

Case Study. We perform a case study to provide a better understanding of the STRUCTADAPT's performance. Table 4 shows an AMR graph in PENMAN notation containing reentrancies (marked in bold) and sentences generated by FINE-TUNE and STRUCTADAPT trained on the LDC2017T10 full training set and in a low-data setting where the models are trained with 2000 data points. FINE-TUNE fails in generating a sentence with the correct concept *break-up* whereas STRUCTADAPT correctly generates a sentence that describes the input graph. The incorrect verb tense is due to lack of tense information in AMR. FINE-TUNE-2000 mixes the semantic relation between *I* and *son* (i.e., mistranslation of the edges in the graph) whereas STRUCTADAPT-2000 generates a correct sentence (except by generating the number 8). Overall, STRUCTADAPT produces a more accurate text output than FINE-TUNE by generating correct pronouns and mentions when control verbs and reentrancies are involved, in both full and low-data scenarios.

Model Variations. In Table 5, we report an ablation study on the impact of distinct adapter components, using adapters only in the encoder or decoder. We evaluate different architecture configurations keeping the same number of parameters for a fair comparison. We find that only training adapters in

	BLEU	BERT
FINE-TUNE	38.5	95.6
ADAPT ONLY ENC	38.5	95.7
ADAPT ONLY DEC	11.6	90.3
ADAPT ENC + DEC	38.6	95.6
STRUCTADAPT-GCN ONLY ENC	40.3	95.9
STRUCTADAPT-GCN ENC + DEC	41.7	96.0

Table 5: Impact of the adapter modules in the encoder or decoder in the LDC2017T10 dev set. All adapter-based models have the same number of parameters.

Linearization	Graph Representation	BLEU	BERT
(i) only nodes	*rep1*	39.1	95.8
	rep2	38.5	95.6
	rep3	38.9	95.7
(ii) nodes and edges	*rep1*	41.7	96.0
	rep2	40.4	95.8
	rep3	40.8	95.9
	complete graph	39.4	95.8

Table 6: Performance on the LDC2017T10 dev set when using different graph representation strategies.

the decoder is not sufficient for a good performance, even having the same number of parameters. This suggests that adapting the PLM encoder to handle graph structures is key in AMR-to-text tasks. Interestingly, the model that only employs STRUCTADAPT in the encoder (i.e., no ADAPT is used in the decoder) has a better performance (+1.7 BLEU) than using ADAPT in both encoder and decoder, highlighting STRUCTADAPT's strong graph encoding abilities. Finally, the best performance is achieved when we employ STRUCTADAPT in the encoder and ADAPT in the decoder, reaching 41.7 BLEU points.

6 Graph Representation Evaluation

In this section, we explore how different graph properties impact the models' abilities to encode the input graph structure.

6.1 Impact of the Graph Representation

Inspired by Damonte and Cohen (2019), we investigate two different approaches when linearizing the AMR: (i) only nodes have explicit representations, whereas edge relations are represented by the adapter parameters using the RGCN;[10] and (ii) the sequence of nodes and edges using depth-first traversal of the graph.

We also propose and evaluate three different graph structures based on subwords (cf. §4.2): *rep1*: for each edge, we connect every token from the source node to every token of the target node; *rep2*: we connect the last token of the source node to the first token of the target node and connect the tokens of a node sequentially; *rep3*: we connect the first token of the source node to the first token of the target node and connect the token of a node sequentially. Figure 3 shows an example of the three representations for an AMR graph structure.

[10]We use regularization based on the basis decomposition for relation weights (Schlichtkrull et al., 2018) since AMR can contain around 150 different edge types.

Additionally, we also investigate a fully connected graph structure (*complete graph*), that is, similarly to the self-attention mechanism in Transformers, all nodes and edges are connected.

As shown in Table 6, explicitly considering nodes and edges in the graph linearization is beneficial. This approach has the advantage of allowing the model to handle new edge relations during inference, as they are not encoded as model parameters. Note that the *complete graph* representation has relatively inferior performance, again demonstrating the advantage of explicitly encoding the input graph connectivity.

Finally, we observe that the best configuration is using nodes and edges with *rep1* (see an example in Figure 3c). We believe that this is because *rep1* allows direct interactions between all source and target tokens, making all token representations of an AMR node directly influenced by the neighbouring tokens.

6.2 Robustness to Graph Linearization

A critical advantage of modeling the graph structure is to be less dependent on linearization strategies because the graph connectivity is invariant to the graph linearization. We thus are interested in measuring the impact of the graph linearization in the models.

Following Hoyle et al. (2021), we investigate three different graph linearizations: (i) CANON: the original order of the canonical human-created linearizations in AMR corpora; (ii) RECONF: the order from the canonical graph linearization is ignored, except for the top node;[11] and (iii) RANDOM: constructs a linearization from a random node in the graph, disregarding all order information from the canonical format, but it remains a valid traversal of the graph. All linearizations are converted to a

[11]RECONF can significantly modify the linearization, including shifting edge labels (e.g., poss to poss-of).

	CANON	RECONF	RANDOM
FINE-TUNE	38.0	35.6	31.3
ADAPT	+0.9	+0.8	+0.9
STRUCTADAPT-RGCN	**+4.1**	**+3.6**	**+5.9**

Table 7: Differences, with respect to FINE-TUNE, in the BLEU score of the LDC2017T10 test set as a function of different graph linearizations.

sequence of node and edge labels using depth-first traversal and used for both training and evaluation. Examples of such graph linearizations are shown in Appendix C.

Table 7 presents the results. Note that while RECONF has a negative impact on all models, STRUCTADAPT has the best performance. ADAPT has similar performance gains over FINE-TUNE in all graph linearizations. Finally, note that for RANDOM, there is a drastic performance drop in FINE-TUNE and the gap between STRUCTADAPT and FINE-TUNE is widest (+5.9 BLEU), demonstrating that explicitly encoding the graph structure is beneficial and that STRUCTADAPT is much less impacted by different graph linearizations.

6.3 Graph Properties

Table 8 shows the effects of the graph size, graph diameter and reentrancies in the performance. First, note that the BLEU scores decrease as the graph size increases since larger graphs often are more complex. The performance gap between STRUCTADAPT and FINE-TUNE becomes larger for relatively larger graphs, showing that STRUCTADAPT is able to better encode complex graphs. As ADAPT is not aware of the graph connectivity, it has much worse scores compared to STRUCTADAPT, especially for larger graphs.

It is expected that the benefit of the STRUCTADAPT will be more evident for AMR graphs containing larger diameter as the encoder is aware of the input graph structure. As seen in Table 8, similarly to the graph size, the scores decrease as the graph diameter increases. STRUCTADAPT achieves a clear improvement when handling graphs with ≥20 diameter, with a improvement of +4.2 BLEU points over FINE-TUNE.

Previous work (Damonte and Cohen, 2019; Szubert et al., 2020) showed that reentrancies (nodes with multiple parents) pose difficulties in encoding AMRs correctly. Because STRUCTADAPT is the only approach to model reentrancies explicitly, we expect it to deal better with these structures. The gap between STRUCTADAPT and the other models is widest for examples with more reentrancies, confirming our hypothesis. In particular, when graphs contain ≥4 reentrancies, STRUCTADAPT has an improvement of +3.6 BLEU points compared to ADAPT.

graph size	1-30	31-60	>60
# **datapoints**	548	537	286
FINE-TUNE	40.6	37.3	38.1
ADAPT	+0.5	+1.4	+1.1
STRUCTADAPT-RGCN	**+2.3**	**+4.0**	**+4.6**
graph diameter	1-10	11-20	>20
# **datapoints**	384	769	218
FINE-TUNE	43.3	37.6	38.5
ADAPT	-0.1	+1.7	+0.3
STRUCTADAPT-RGCN	**+0.5**	**+4.3**	**+4.2**
# **reentrancies**	0	1-3	4-20
# **datapoints**	619	664	88
FINE-TUNE	42.9	38.0	31.3
ADAPT	+0.2	+1.7	+0.8
STRUCTADAPT-RGCN	**+3.4**	**+4.4**	**+4.4**

Table 8: Differences, with respect to FINE-TUNE, in the BLEU score of the LDC2017T10 test set as a function of the graph size, graph diameter and number of reentrancies.

7 Conclusion

We presented STRUCTADAPT, a novel adapter architecture to explicitly model graph structures into pretrained language models, providing an extensive evaluation of our approach and showing that it achieves state-of-the-art results on two AMR-to-text benchmarks, training much fewer parameters. We also found that STRUCTADAPT is more effective when encoding complex graphs, when trained on fewer datapoints, and is more robust to different graph linearizations and reentrancies. In future work, we plan to consider other graph-to-text tasks, such as those based on Knowledge Graphs.

Acknowledgments

We thank our anonymous reviewers for their thoughtful comments. We also would like to thank Jonas Pfeiffer, Jorge Cardona, Juri Opitz, Kevin Stowe, Thy Tran, Tilman Beck and Tim Baumgärtner for their feedback on this work. This work has been supported by the German Research Foundation (DFG) as part of the Research Training Group "Adaptive Preparation of Information form Heterogeneous Sources" (AIPHES, GRK 1994/1) and as part of the DFG funded project UKP-SQuARE with the number GU 798/29-1.

References

Xuefeng Bai, Linfeng Song, and Yue Zhang. 2020. Online back-parsing for AMR-to-text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1206–1219, Online. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

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

Michele Bevilacqua, Rexhina Blloshmi, and Roberto Navigli. 2021. One spring to rule them both: Symmetric amr semantic parsing and generation without a complex pipeline. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(14):12564–12573.

Deng Cai and Wai Lam. 2020. Graph transformer for graph-to-sequence learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7464–7471.

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

Marco Damonte and Shay B. Cohen. 2019. Structural neural encoders for AMR-to-text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3649–3658, Minneapolis, Minnesota. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016. Generation from Abstract Meaning Representation using tree transducers. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 731–739, San Diego, California. Association for Computational Linguistics.

Qiankun Fu, Linfeng Song, Wenyu Du, and Yue Zhang. 2021. End-to-end AMR corefencence resolution. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4204–4214, Online. Association for Computational Linguistics.

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

Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. 2014. An empirical investigation of catastrophic forgeting in gradient-based neural networks. In *Proceedings of International Conference on Learning Representations (ICLR)*.

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

Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. 2021. WARP: Word-level Adversarial ReProgramming. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4921–4933, Online. Association for Computational Linguistics.

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

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings*

of Machine Learning Research, pages 2790–2799. PMLR.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with MACE. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1120–1130, Atlanta, Georgia. Association for Computational Linguistics.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Alexander Miserlis Hoyle, Ana Marasović, and Noah A. Smith. 2021. Promoting graph awareness in linearized graph-to-text generation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 944–956, Online. Association for Computational Linguistics.

Mihir Kale. 2020. Text-to-text pre-training for data-to-text tasks. *arXiv e-prints*.

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

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157, Vancouver, Canada. Association for Computational Linguistics.

Anne Lauscher, Olga Majewska, Leonardo F. R. Ribeiro, Iryna Gurevych, Nikolai Rozanov, and Goran Glavaš. 2020. Common sense or world knowledge? investigating adapter-based knowledge injection into pretrained transformers. In *Proceedings of Deep Learning Inside Out (DeeLIO): The First Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 43–49, Online. Association for Computational Linguistics.

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

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

Kexin Liao, Logan Lebanoff, and Fei Liu. 2018. Abstract Meaning Representation for multi-document summarization. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1178–1190, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. GPT understands, too. *CoRR*, abs/2103.10385.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Roberta: A robustly optimized bert pretraining approach. *arXiv e-prints*.

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

Linyong Nan, Dragomir Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, Yangxiaokang Liu, Nadia Irwanto, Jessica Pan, Faiaz Rahman, Ahmad Zaidi, Mutethia Mutuma, Yasin Tarabar, Ankit Gupta, Tao Yu, Yi Chern Tan, Xi Victoria Lin, Caiming Xiong, Richard Socher, and Nazneen Fatema Rajani. 2021. 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, Online. Association for Computational Linguistics.

Juri Opitz, Angel Daza, and Anette Frank. 2021. Weisfeiler-leman in the bamboo: Novel amr graph metrics and a benchmark for amr graph similarity. *Transactions of the Association for Computational Linguistics*.

Juri Opitz and Anette Frank. 2021. Towards a decomposable metric for explainable evaluation of text generation from AMR. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1504–1518, Online. Association for Computational Linguistics.

Juri Opitz, Letitia Parcalabescu, and Anette Frank. 2020. AMR similarity metrics from principles. *Transactions of the Association for Computational Linguistics*, 8:522–538.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. ToTTo: A controlled table-to-text generation dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. AdapterFusion: Non-Destructive Task Composition for Transfer Learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020a. AdapterHub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 46–54, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020b. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online. Association for Computational Linguistics.

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

Nima Pourdamghani, Kevin Knight, and Ulf Hermjakob. 2016. Generating English from Abstract Meaning Representations. In *Proceedings of the 9th International Natural Language Generation conference*, pages 21–25, Edinburgh, UK. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. In *Technical report, OpenAI*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2017. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems*, volume 30, pages 506–516. Curran Associates, Inc.

Leonardo F. R. Ribeiro, Claire Gardent, and Iryna Gurevych. 2019. Enhancing AMR-to-text generation with dual graph representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3183–3194, Hong Kong, China. Association for Computational Linguistics.

Leonardo F. R. Ribeiro, Jonas Pfeiffer, Yue Zhang, and Iryna Gurevych. 2021. Smelting gold and silver for improved multilingual amr-to-text generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Punta Cana, November 7-11, 2021*.

Leonardo F. R. Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. 2020a. Investigating pretrained language models for graph-to-text generation. *arXiv e-prints*.

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

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, pages 593–607.

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

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational*

Linguistics (Volume 1: Long Papers), pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Linfeng Song, Daniel Gildea, Yue Zhang, Zhiguo Wang, and Jinsong Su. 2019. Semantic neural machine translation using amr. *Transactions of the Association for Computational Linguistics*, 7:19–31.

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

Ida Szubert, Marco Damonte, Shay B. Cohen, and Mark Steedman. 2020. The role of reentrancies in Abstract Meaning Representation parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2198–2207, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Pavlos Vougiouklis, Hady Elsahar, Lucie-Aimée Kaffee, Christophe Gravier, Frédérique Laforest, Jonathon Hare, and Elena Simperl. 2018. Neural wikipedian: Generating textual summaries from knowledge base triples. *Journal of Web Semantics*, 52-53:1 – 15.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In H. Wallach, H. Larochelle, A. Beygelzimer, F. Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 3266–3280.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

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

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Jeffrey O. Zhang, Alexander Sax, Amir Zamir, Leonidas J. Guibas, and Jitendra Malik. 2019. Side-tuning: Network adaptation via additive side networks. *arXiv e-prints*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020a. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Yan Zhang, Zhijiang Guo, Zhiyang Teng, Wei Lu, Shay B. Cohen, Zuozhu Liu, and Lidong Bing. 2020b. Lightweight, dynamic graph convolutional networks for AMR-to-text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2162–2172, Online. Association for Computational Linguistics.

Appendices

In this supplementary material, we detail experiments' settings and additional information about the human evaluation and graph representations.

A Details of Models and Hyperparameters

The experiments were executed using the version 3.3.1 of the *transformers* library released by Hugging Face (Wolf et al., 2019). In Table 9, we report the hyperparameters used to train the models presented in this paper. We train until the development set BLEU has not improved for 5 epochs.

	learning rate	batch size	beam search size
FINE-TUNE	3e-05	4	5
FT-TOP2	1e-04	4	5
FT-BOTTOM2	1e-04	4	5
ADAPT	1e-04	4	5
STRUCTADAPT	1e-04	4	5

Table 9: Hyperparameter settings for our methods.

B Details on the Human Evaluation

The human evaluation was conducted via Amazon Mechanical Turk. We randomly select 100 generated texts for each of the 3 sets and each adapter model (ADAPT, STRUCTADAPT-GCN), with a total of 600 texts to be evaluated. The annotators then rate the meaning similarity on a 1-7 Likert scale. For each text, we collect scores from 3 annotators. We use MACE (Hovy et al., 2013) to further improve upon these raw answers by unsupervised estimation of worker trustworthiness and subsequent recovery of the most likely score. Models are ranked according to the mean of sentence-level scores. We defined a filter for all our evaluations, allowing to participate only workers who have more than 5000 HITs approved and with an acceptance rate of 95% or higher. The task took workers a median time of 1.6 minutes per pair of sentences. We apply a quality control step filtering workers who do not score some faked and known sentences properly or did the experiment in a very short time.

C Example of Graph Linearizations

In Table 10, we present three different linearizations for the same AMR graph and its corresponding reference sentence. Figure 5 shows the two possible graphs that are represented by the linearizations. In particular, Figure 5a shows a graph that is represented by CANON and RECONF linearizations and Figure 5b shows a graph that is represented by RANDOM. Note that whereas the linearizations can greatly differ from each other, the graph structure for all linearizations remains very similar.

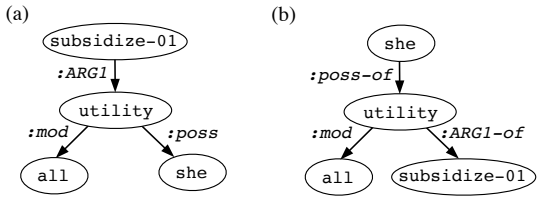


Figure 5: Two AMR graphs with the same meaning.

CANON

```
(s / subsidize-01
    :ARG1 (u / utility
        :poss (s2 / she)
        :mod (a / all)))
```

RECONF

```
(s / subsidize-01
    :ARG1 (u / utility
        :mod (a / all)
        :poss (s2 / she)))
```

RANDOM

```
(s2 / she
    :poss-of (u / utility
        :ARG1-of (s / subsidize-01)
        :mod (a / all)))
```

SENTENCE: Her utilities are all subsidized.

Table 10: Different linearizations for an AMR graph.

Chapter 9

Smelting Gold and Silver for Improved Multilingual AMR-to-Text Generation

Smelting Gold and Silver for Improved Multilingual AMR-to-Text Generation

Leonardo F. R. Ribeiro[†], **Jonas Pfeiffer**[†], **Yue Zhang**[‡] **and Iryna Gurevych**[†]

[†]Ubiquitous Knowledge Processing Lab, Technical University of Darmstadt
[‡]School of Engineering, Westlake University
ribeiro@aiphes.tu-darmstadt.de

Abstract

Recent work on multilingual AMR-to-text generation has exclusively focused on data augmentation strategies that utilize silver AMR. However, this assumes a high quality of generated AMRs, potentially limiting the transferability to the target task. In this paper, we investigate different techniques for automatically generating AMR annotations, where we aim to study which source of information yields better multilingual results. Our models trained on gold AMR with silver (machine translated) sentences outperform approaches which leverage generated silver AMR. We find that combining both complementary sources of information further improves multilingual AMR-to-text generation. Our models surpass the previous state of the art for German, Italian, Spanish, and Chinese by a large margin.[1]

1 Introduction

AMR-to-text generation is the task of recovering a text with the same meaning as a given Abstract Meaning Representation (AMR) (Banarescu et al., 2013), and has recently received much research interest (Ribeiro et al., 2019; Wang et al., 2020; Mager et al., 2020; Harkous et al., 2020; Fu et al., 2021). AMR has applications to a range of NLP tasks, including summarization (Hardy and Vlachos, 2018) and spoken language understanding (Damonte et al., 2019), and has the potential power of acting as an *interlingua* that allows the generation of text in many different languages (Damonte and Cohen, 2018; Zhu et al., 2019).

While previous work has predominantly focused on monolingual English settings (Cai and Lam, 2020b; Bevilacqua et al., 2021), recent work has also studied multilinguality in meaning representations (Blloshmi et al., 2020; Sheth et al., 2021). Whereas Damonte and Cohen (2018) demonstrate that parsers can be effectively trained to transform multilingual text into English AMR, Mille et al. (2018, 2019) and Fan and Gardent (2020) discuss the reverse task, turning meaning representations into multilingual text, as shown in Figure 1. However, gold-standard multilingual AMR training data is currently scarce, and previous work (Fan and Gardent, 2020) while discussing the feasibility of multilingual AMR-to-text generation, has investigated synthetically generated AMR as the *only source* of silver training data.

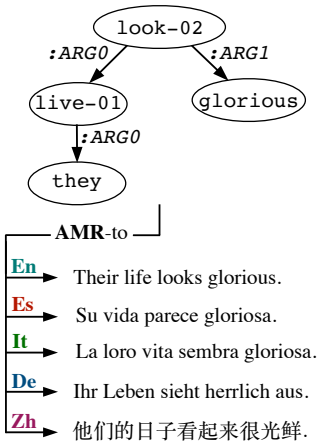


Figure 1: A generation example from English AMR to multiple different languages.

In this paper, we aim to close this gap by providing an extensive analysis of different augmentation techniques to cheaply acquire silver-standard multilingual AMR-to-text data: (1) Following Fan and Gardent (2020), we parse English sentences into silver AMRs from parallel multilingual corpora (SILVERAMR), resulting in a dataset consisting of grammatically correct sentences with noisy AMR structures. (2) We leverage machine translation (MT) and translate the English sentences from the gold AMR-to-text corpus to the respective target languages (SILVERSENT), resulting in a dataset with correct AMR structures but potentially unfaithful or non-grammatical sentences. (3) We experiment

[1]Our code and checkpoints are available at https://github.com/UKPLab/m-AMR2Text.

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 742–750
November 7–11, 2021.

with utilizing the AMR-to-text corpus with both gold English AMR and sentences in multi-source scenarios to enhance multilingual training.

Our contributions and the organization of this paper are the following: First, we formalize the multilingual AMR-to-text generation setting and present various cheap and efficient alternatives for collecting multilingual training data. Second, we show that our proposed training strategies greatly advance the state of the art finding that SILVERSENT considerably outperforms SILVERAMR. Third, we show that SILVERAMR has better relative performance in relatively larger sentences, whereas SILVERSENT performs better for relatively larger graphs. Overall, we find that a combination of both strategies further improves the performance, showing that they are complementary for this task.

2 Related Work

Approaches for AMR-to-text generation predominantly focus on English, and typically employ an encoder-decoder architecture, employing a linearized representation of the graph (Konstas et al., 2017; Ribeiro et al., 2020a). Recently, models based on the graph-to-text paradigm (Ribeiro et al., 2020b; Schmitt et al., 2021) improve over linearized approaches, explicitly encoding the AMR structure with a graph encoder (Song et al., 2018; Beck et al., 2018; Ribeiro et al., 2019; Guo et al., 2019; Cai and Lam, 2020b; Ribeiro et al., 2021).

Advances in multilingual AMR parsing have focused on a variety of different languages such as Brazilian Portuguese, Chinese, Czech and Spanish (Hajič et al., 2014; Xue et al., 2014; Migueles-Abraira et al., 2018; Sobrevilla Cabezudo and Pardo, 2019). In contrast, little work has focused on the reverse AMR-to-text setting (Fan and Gardent, 2020). We aim to close this gap by experimenting with different data augmentation methods for efficient multilingual AMR-to-text generation.

3 Multilingual AMR-to-Text Generation

In AMR-to-text generation, we transduce an AMR graph $\mathcal{G}$ to a surface realization as a sequence of tokens $y = \langle y_1, \ldots, y_{|y|} \rangle$. As input we use an English-centric AMR graph where the output y can be realized in different languages (see Figure 1).

3.1 Approach

We employ mT5 (Xue et al., 2021), a Transformer-based encoder-decoder architecture (Vaswani et al., 2017), motivated by prior work (Ribeiro et al., 2020a, 2021) that leverages T5 (Raffel et al., 2019) for AMR-to-text generation.

We define $x = \text{LIN}(\mathcal{G})$, where LIN is a function that linearizes $\mathcal{G}$ into a sequence of node and edge labels using depth-first traversal of the graph (Konstas et al., 2017). x is encoded, conditioned on which the decoder predicts y autoregressively.

Consequently, the encoder is required to learn language agnostic representations amenable to be used in a multilingual setup for the English AMR graph; the decoder attends over the encoded AMR and is required to generate text in different languages with varied word order and morphology.

To differentiate between languages, we prepend a prefix "`translate AMR to <tgt_language>:`" to the AMR graph representation.[2] We add the edge labels which are present in the AMR graphs of the LDC2017T10 training set to the encoder's vocabulary in order to avoid considerable subtoken splitting – this allows us to encode the AMR with a compact sequence of tokens and also learn explicit representations for the AMR edge labels. Finally, this multilingual approach allows us to have more AMR data on the encoder side when increasing the number of considered languages. This could be particularly helpful when using languages with little training data.

3.2 Data

Since gold-standard *training* data for multilingual AMR-to-text generation does not exist, data augmentation methods are necessary. Given a set of gold AMR training data for English and parallel corpora between English and target languages, we thus aim to identify the best augmentations strategies to achieve multilingual generation.

As our monolingual AMR-to-text training dataset, we consider the LDC2017T10 dataset (GOLDAMR), containing English AMR graphs and sentences. We evaluate our different approaches on the multilingual LDC2020T07 test set by Damonte and Cohen (2018) consisting of gold annotations for Spanish (ES), Italian (IT), German (DE) and Chinese (ZH).[3] For our multilingual parallel sentence corpus we consider data from different sources. For ES, IT and DE, we use: **Europarl-v7** (Koehn, 2005), an aligned corpus of European Union parlia-

[2] For example, for AMR-to-Spanish we use the prefix "`translate AMR to Spanish:`".

[3] This dataset was constructed by professional translators based on the LDC2017T10 test set.

	BLEU					BERTscore				
	ES	IT	DE	ZH	All	ES	IT	DE	ZH	All
MT (Fan and Gardent, 2020)	21.6	19.6	15.7	-	-	-	-	-	-	-
Multilingual model (Fan and Gardent, 2020)	21.7	19.8	15.3	-	-	-	-	-	-	-
MT	27.6	24.2	19.4	23.3	23.6	87.1	85.7	83.5	79.9	84.0
SILVERAMR	23.3	21.2	16.9	20.1	20.4	84.5	83.7	82.0	76.3	81.6
SILVERSENT	28.3	24.3	18.9	22.2	23.4	87.3	85.7	83.5	79.6	84.0
SILVERAMR + GOLDAMR	28.2	24.9	19.4	22.9	23.9	87.6	85.9	83.9	79.5	84.2
SILVERSENT + GOLDAMR	28.5	24.6	19.2	22.3	23.7	87.3	85.8	83.6	79.6	84.0
SILVERAMR + SILVERSENT	**30.7**	**26.4**	**20.6**	**24.2**	**25.5**	87.8	**86.3**	**84.1**	**80.5**	**84.7**
SILVERAMR + SILVERSENT + GOLDAMR	30.4	26.1	20.5	23.4	25.1	**88.0**	**86.3**	**84.1**	80.1	84.6

Table 1: Results on the multilingual LDC2020T07 test set. When training on multiple seeds, the standard deviation is between 0.1 an 0.3 BLEU. The results of our models compared to the MT baseline are statistically significant.

mentary debates; **Tatoeba**,[4] a large database of example sentences and translations; and **TED2020**,[5] a dataset of translated subtitles of TED talks. For ZH, we use the **UM-Corpus** (Tian et al., 2014).

3.3 Creating Silver Training Data

We experiment with two augmentation techniques that generate silver-standard multilingual training data, described in what follows.

SILVERAMR. We follow Fan and Gardent (2020) and leverage the multilingual parallel corpora described in §3.2 and generate AMRs for the respective English sentences.[6] While the multilingual sentences are of gold standard, the AMR graphs are of silver quality. Similar to Fan and Gardent (2020), for each target language we extract a parallel dataset of 1.9M sentences.

SILVERSENT. We fine-tune mT5 as a translation model for English to the respective target languages, using the same parallel sentences used in SILVERAMR. Then, we translate the English sentences of GOLDAMR into the respective target languages, resulting in a multilingual dataset that consists of gold AMRs and silver sentences. The multilingual training dataset contains 36,521 examples for each target language.

4 Experiments

We implement our models using mT5_{base} from HuggingFace (Wolf et al., 2020). We use the Adafactor optimizer (Shazeer and Stern, 2018) and employ a linearly decreasing learning rate schedule without warm-up. The hyperparameters we tune include the batch size, number of epochs and learning rate.[7] The models are evaluated in the multilingual LDC2020T07 test set, using BLEU (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2014), chrF++ (Popović, 2015) and BERTscore (Zhang et al., 2020) metrics. We compare with a MT baseline – we generate the test set with an AMR-to-English model trained with T5 (Ribeiro et al., 2021) and translate the generated English sentences to the target language using MT. For a fair comparison, our MT model is based on mT5 and trained with the same data as the other approaches.

Training Strategies. We propose different training strategies under the setting of §3.2 in order to investigate which combination leads to stronger multilingual AMR-to-text generation. Besides training models using SILVERAMR or SILVERSENT, we investigate different combinations of multi-source training also using GOLDAMR.

Main Results. Table 1 shows our main results.[8] First, SILVERAMR substantially outperforms Fan and Gardent (2020) despite being trained on the same amount of silver AMR data. We believe this is because we utilize mT5, whereas Fan and Gardent (2020) use XLM (Conneau et al., 2020), and our parallel data may contain different domain data.

SILVERSENT considerably outperforms SILVERAMR in all metrics, despite SILVERAMR consisting of two orders of magnitude more data. We believe the reasons are twofold: Firstly, the correct semantic structure of gold AMR annotations is necessary to learn a faithful realization; Secondly, SILVERSENT provides examples of the same domain as the evaluation test set. We observe similar performance to SILVERSENT when training on both GOLDAMR and SILVERAMR, indicating that the combination of target domain data and gold AMR graphs are

[4] https://tatoeba.org/

[5] https://github.com/UKPLab/sentence-transformers/tree/master/docs/datasets

[6] The English sentences of the parallel corpus are parsed using a state-of-the-art AMR parser (Cai and Lam, 2020a).

[7] Hyperparameter details are in the appendix A.

[8] METEOR and chrF++ results can be found in Appendix Table 6.

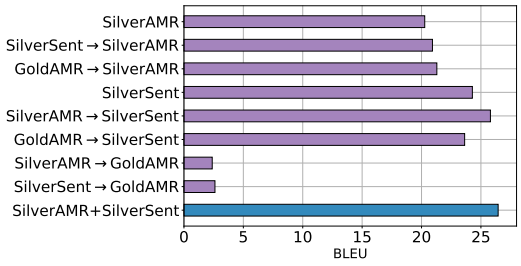


Figure 2: Order impact of sequential fine-tuning for IT.

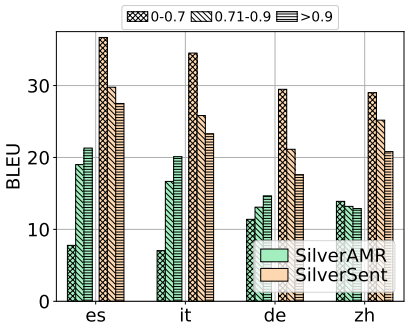


Figure 3: Impact of the sentence length and graph size ratio γ on the LDC2020T07 multilingual test set.

	ES	IT	DE	ZH
SILVERAMR	19.3	16.5	11.8	11.9
SILVERSENT	22.3	17.3	12.7	11.9
SILVERAMR + SILVERSENT	23.5	19.2	15.0	13.0

Table 2: BLEU results for out of domain evaluation.

necessary for downstream task performance. However, training on both GOLDAMR and SILVERSENT yields small gains, indicating that the respective information is adequately encoded within the silver standard dataset.

We observe similar patterns when combining the silver standard datasets. While SILVERAMR+SILVERSENT complement each other, resulting in the overall best performance, adding GOLDAMR does not yield any notably gains. These results demonstrate that *both* gold AMR structure and gold sentence information are important for training multilingual AMR-to-text models, while SILVERSENT are seemingly more important.

Effect of the Fine-tuning Order. In Figure 2 we illustrate the impact of different data source orderings when fine-tuning in a two-phase setup for IT.[9] Firstly, we observe a decrease in performance for all sequential fine-tuning settings, compared to our proposed mixed multi-source training, which is likely due to *catastrophic forgetting*.[10] Secondly, training on SILVERAMR and subsequently on SILVERSENT (or vice versa), improves performance over only using either, again demonstrating their complementarity. Thirdly, SILVERSENT continues to outperform SILVERAMR as a second task. Finally, GOLDAMR is not suitable as the second task for multilingual settings as the model predominantly generates English text.

Impact of Sentence Length and Graph Size. As silver annotations potentially lead to noisy inputs, models trained on SILVERAMR are potentially less capable of encoding the AMR semantics correctly, and models trained on SILVERSENT potentially generate fluent sentences less reliably. To analyze the advantages of the two forms of data, we measure the performance against the sentence lengths and graph sizes.[11] We define γ to be a ratio of the sentence length, divided by the number of AMR graph nodes. In Figure 3 we plot the respective results for SILVERAMR and SILVERSENT, categorized into three bins. We find that almost all SILVERAMR's BLEU increases for longer sentences, suggesting that training with longer gold *sentences* improves performance. In contrast, with larger *graphs*, the BLEU performance improves for SILVERSENT, indicating that large gold AMR graphs are also important. SILVERAMR and SILVERSENT present relative gains in performance on opposite ratios of sentence length and graph size, suggesting that they capture distinct aspects of the data.

Out of Domain Evaluation. To disentangle the effects of in-domain sentences and gold quality AMR graphs in SILVERSENT, we evaluate both silver data approaches on the **Weblog and WSJ** subset of the LDC2020T07 dataset; The domain of this subset is *not included* in the LDC2017T10 training set. We present the BLEU results in Table 2.[12] While we find that SILVERSENT prevails in achieving better performance — demonstrating that AMR gold structures are an important source for training multilingual AMR-to-text models — SILVERAMR and SILVERSENT perform more comparably than when evaluated on the full LDC2020T07 test set. This demonstrates that the domain transfer factor plays an important role in the strong performance of SILVERSENT. Overall, SILVERAMR+SILVERSENT outperforms both single source settings, establishing the

[9] Other languages follow similar trends and are presented in Figure 4 in the Appendix.

[10] The model trained on the second task forgets the first task.

[11] Sentence lengths were measured using subwords.

[12] BERTscore results can be found in Appendix Table 5.

Model	Examples
AMR	(m / multi-sentence :snt1 (w2 / wish-01 :ARG0 (i2 / i) :ARG1 (p / possible-01 :ARG1 (w3 / wipe-out-02 :ARG0 i2 :ARG1 (s / she) :source (l / live-01 :ARG0 i2)))) :snt2 (g / good-02 :ARG1 (t / thing) :degree (m2 / more :degree (m3 / much :degree (s2 / so))) :prep-without (s3 / she)))
SILVERAMR	**Con ella, las cosas son mucho mejor**. Deseo que pudiera eliminarla de mi vida.
SILVERSENT	Desearía **que podía** eliminarla de mi vida. Las cosas serían mucho mejor sin ella.
SILVERAMR+SILVERSENT	Desearía poder eliminarla de mi vida, las cosas serían mucho mejor sin ella.
Reference	Ojalá pudiera borrarla de mi vida, las cosas hubieran sido mucho mejor sin ella.
English Reference	I wish I could wipe her out of my life - things would be so much better without her.

Table 3: Example of an AMR, generated texts in ES by the different models, and its ES and EN references. We indicate in **red** errors (unfaithfulness in SILVERAMR and incorrect grammar in SILVERSENT) that are not present in SILVERAMR+SILVERSENT and in the human-written reference.

complementarity of both silver sources of data.

Case Study. Table 3 shows an AMR, its reference sentences in ES and EN, and sentences generated in ES by SILVERAMR, SILVERSENT, and their combination. The incorrect verb tense is due to the lack of tense information in AMR. SILVERAMR fails in capturing the correct concept *prep-without* generating an unfaithful first sentence. This demonstrates a potential issue with approaches trained with silver AMR data where the input graph structure can be noisy, leading to a model less capable of encoding AMR semantics. On the other hand, SILVERSENT correctly generates sentences that describe the graph, while it still generates a grammatically incorrect sentence (wrongly generating *que podía* after *desearía*). This highlights a potential problem with approaches that employ silver sentence data where sentences used for the training could be ungrammatical, leading to models less capable of generating a fluent sentence. Finally, SILVERAMR+SILVERSENT produces a more accurate output than both silver approaches by generating grammatically correct and fluent sentences, correct pronouns, and mentions when control verbs and reentrancies (nodes with more than one entering edge) are involved.

5 Conclusion

The unavailability of gold training data makes multilingual AMR-to-text generation a challenging topic. We have extensively evaluated data augmentation methods by leveraging existing resources, namely a set of gold English AMR-to-text data and a corpus of multilingual parallel sentences. Our experiments have empirically validated that both sources of silver data — silver AMR with gold sentences and gold AMR with silver sentences — are complementary, and a combination of both leads to state-of-the-art performance on multilingual AMR-to-text generation tasks.

Acknowledgments

We would like to thank Gözde Gül Sahin, Ji-Ung Lee, Kevin Stowe, Kexin Wang and Nandan Thakur for their feedback on this work. Leonardo F. R. Ribeiro is supported by the German Research Foundation (DFG) as part of the Research Training Group "Adaptive Preparation of Information form Heterogeneous Sources" (AIPHES, GRK 1994/1) and as part of the DFG funded project UKP-SQuARE with the number GU 798/29-1. Jonas Pfeiffer is supported by the LOEWE initiative (Hesse, Germany) within the emergenCITY center.

References

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

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

Michele Bevilacqua, Rexhina Blloshmi, and Roberto Navigli. 2021. One spring to rule them both: Symmetric amr semantic parsing and generation without a complex pipeline. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(14):12564–12573.

Rexhina Blloshmi, Rocco Tripodi, and Roberto Navigli. 2020. XL-AMR: Enabling cross-lingual AMR parsing with transfer learning techniques. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2487–2500, Online. Association for Computational Linguistics.

Deng Cai and Wai Lam. 2020a. AMR parsing via graph-sequence iterative inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1290–1301, Online. Association for Computational Linguistics.

Deng Cai and Wai Lam. 2020b. Graph transformer for graph-to-sequence learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7464–7471.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Marco Damonte and Shay B. Cohen. 2018. Cross-lingual Abstract Meaning Representation parsing. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1146–1155, New Orleans, Louisiana. Association for Computational Linguistics.

Marco Damonte, Rahul Goel, and Tagyoung Chung. 2019. Practical semantic parsing for spoken language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Industry Papers)*, pages 16–23, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Angela Fan and Claire Gardent. 2020. Multilingual AMR-to-text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2889–2901, Online. Association for Computational Linguistics.

Qiankun Fu, Linfeng Song, Wenyu Du, and Yue Zhang. 2021. End-to-end AMR corefencence resolution. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4204–4214, Online. Association for Computational Linguistics.

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

Jan Hajič, Ondřej Bojar, and Zdeňka Urešová. 2014. Comparing Czech and English AMRs. In *Proceedings of Workshop on Lexical and Grammatical Resources for Language Processing*, pages 55–64, Dublin, Ireland. Association for Computational Linguistics and Dublin City University.

Hardy Hardy and Andreas Vlachos. 2018. Guided neural language generation for abstractive summarization using Abstract Meaning Representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 768–773, Brussels, Belgium. Association for Computational Linguistics.

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

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand. AAMT, AAMT.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157, Vancouver, Canada. Association for Computational Linguistics.

Manuel Mager, Ramón Fernandez Astudillo, Tahira Naseem, Md. Arafat Sultan, Young-Suk Lee, Radu Florian, and Salim Roukos. 2020. Gpt-too: A language-model-first approach for amr-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 1846–1852.

Noelia Migueles-Abraira, Rodrigo Agerri, and Arantza Diaz de Ilarraza. 2018. Annotating Abstract Meaning Representations for Spanish. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Simon Mille, Anja Belz, Bernd Bohnet, Yvette Graham, Emily Pitler, and Leo Wanner. 2018. The first multilingual surface realisation shared task (SR'18): Overview and evaluation results. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 1–12, Melbourne, Australia. Association for Computational Linguistics.

Simon Mille, Anja Belz, Bernd Bohnet, Yvette Graham, and Leo Wanner. 2019. The second multilingual surface realisation shared task (SR'19): Overview and evaluation results. In *Proceedings of the 2nd Workshop on Multilingual Surface Realisation (MSR 2019)*, pages 1–17, Hong Kong, China. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*.

Leonardo F. R. Ribeiro, Claire Gardent, and Iryna Gurevych. 2019. Enhancing AMR-to-text generation with dual graph representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3183–3194, Hong Kong, China. Association for Computational Linguistics.

Leonardo F. R. Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. 2020a. Investigating pretrained language models for graph-to-text generation. *arXiv e-prints*.

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

Leonardo F. R. Ribeiro, Yue Zhang, and Iryna Gurevych. 2021. Structural adapters in pretrained language models for amr-to-text generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Punta Cana, November 7-11, 2021*.

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

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4596–4604. PMLR.

Janaki Sheth, Young-Suk Lee, Ramón Fernandez Astudillo, Tahira Naseem, Radu Florian, Salim Roukos, and Todd Ward. 2021. Bootstrapping multilingual AMR with contextual word alignments. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 394–404, Online. Association for Computational Linguistics.

Marco Antonio Sobrevilla Cabezudo and Thiago Pardo. 2019. Towards a general Abstract Meaning Representation corpus for Brazilian Portuguese. In *Proceedings of the 13th Linguistic Annotation Workshop*, pages 236–244, Florence, Italy. Association for Computational Linguistics.

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

Liang Tian, Derek F. Wong, Lidia S. Chao, Paulo Quaresma, Francisco Oliveira, and Lu Yi. 2014. Um-corpus: A large english-chinese parallel corpus for statistical machine translation. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014, Reykjavik, Iceland, May 26-31, 2014*, pages 1837–1842.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Tianming Wang, Xiaojun Wan, and Hanqi Jin. 2020. AMR-to-text generation with graph transformer. *Transactions of the Association for Computational Linguistics*, 8:19–33.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online, November 16-20, 2020*, pages 38–45. Association for Computational Linguistics.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Nianwen Xue, Ondřej Bojar, Jan Hajič, Martha Palmer, Zdeňka Urešová, and Xiuhong Zhang. 2014. Not an interlingua, but close: Comparison of English AMRs to Chinese and Czech. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1765–1772, Reykjavik, Iceland. European Language Resources Association (ELRA).

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Huaiyu Zhu, Yunyao Li, and Laura Chiticariu. 2019. Towards universal semantic representation. In *Proceedings of the First International Workshop on Designing Meaning Representations*, pages 177–181, Florence, Italy. Association for Computational Linguistics.

Appendices

A Details of Models and Hyperparameters

The experiments were executed using the version 4.4.0 of the *transformers* library by Hugging Face (Wolf et al., 2020). Table 4 shows the hyperparameters used to train our models. BLEU is used for model selection using translated sentences of the LDC2017T10 development set. We train until the results on the development set BLEU have not improved for 6 epochs.

learning rate	1e-04
batch size	8
beam search size	6
max source length	350
max target length	200

Table 4: Hyperparameter settings for our methods.

B Main Results: Additional Metrics

In Table 6 we present additional results on the multilingual LDC2020T07 test set using METEOR (Denkowski and Lavie, 2014), chrF++ (Popović, 2015) metrics.

C Results: Out of Domain Evaluation

In Table 5 we show BERTscore (Zhang et al., 2020) results for out of domain evaluation on the **Weblog and WSJ** subset of the LDC2020T07 dataset.

	ES	IT	DE	ZH
SILVERAMR	83.3	81.2	79.8	73.6
SILVERSENT	84.6	83.0	80.4	73.0
SILVERAMR + SILVERSENT	84.6	83.2	81.2	74.1

Table 5: BERT scores for out of domain evaluation.

D Results: Sequential Fine-tuning

In Figure 4 we present the impact of sequential fine-tuning strategies in the LDC2020T07 test set for ES, DE and ZH.

	METEOR					chrF++				
	ES	IT	DE	ZH	All	ES	IT	DE	ZH	All
MT	29.9	27.2	23.2	25.7	26.5	54.8	52.0	47.3	22.3	44.1
SILVERAMR	28.3	26.0	22.7	23.3	25.0	51.3	49.6	45.9	19.5	41.5
SILVERSENT	30.6	27.3	23.0	24.9	26.4	55.6	52.2	47.2	21.7	44.1
SILVERAMR + GOLDAMR	29.8	26.9	23.6	25.2	26.3	55.9	51.7	47.5	22.3	44.3
SILVERSENT + GOLDAMR	30.4	27.5	23.3	24.9	26.5	55.3	52.3	47.3	21.8	44.1
SILVERAMR + SILVERSENT	**31.9**	**28.7**	**24.4**	**26.4**	**27.8**	**57.2**	**54.0**	**49.4**	**23.0**	**45.9**
SILVERAMR + SILVERSENT + GOLDAMR	31.7	28.6	24.2	25.7	27.5	**57.2**	53.6	48.6	22.5	45.4

Table 6: METEOR and chrF++ results on the multilingual LDC2020T07 test set.

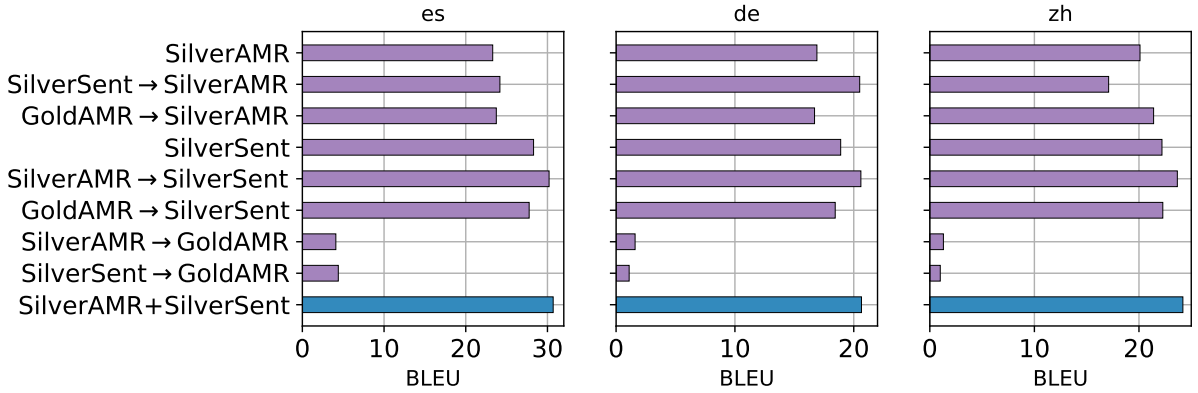


Figure 4: Order impact of sequential fine-tuning in the LDC2020T07 test set for ES, DE and ZH.

Part III

Epilogue

Chapter 10

Conclusion and Future Work

10.1 Conclusion

Graph-to-text generation has the potential to enable AI systems to communicate complex information stored into graph-based inputs to humans in the form best understood by us – natural language. In this thesis, we have focused on the important advances achieved by applying deep learning techniques to this task. We have proposed novel neural approaches for encoding graph-based data and demonstrated that they are able to better represent semantic graph structures leading to improved text generation in different downstream applications. These encoding mechanisms are essential for developing generative natural language models that can deal with complex relational data stored in the form of graphs, and are valuable tools that enable the easy consumption of knowledge for people all over the world.

Our proposed methods are incorporated into several text generation models enabling the capacity to explicitly capture the semantic relations between nodes in the input graph, consequently improving the generated text's fluency and faithfulness. In Chapter 4, we developed a dual graph encoder for AMR-to-text generation that produces parallel top-down and bottom-up representations of nodes capturing contrasting views of the graph and showed that this encoding strategy improves the meaning similarity and readability of the generated sentences. Subsequently, in Chapter 5, we showed a graph-to-text framework based on graph attention networks that provides different configurations to incorporate global and local node aggregation, promoting the generation of more coherent multi-sentence outputs and enhancing their fluency and adequacy. In Chapter 6, we presented a self-attention strategy for injecting graph connectivity into Transformer architecture that learns differently connected views and global patterns extracted from the input graph, enabling better text generation while being parameter efficient. Taken together, these methods are applicable to a variety of other data-to-text tasks and represent complementary approaches that can be used to enhance encoder-decoder neural models in different setups that operate with structured data.

This thesis also addressed the challenges of adapting language models that were

primarily pretrained in plain text to structured data. In Chapter 7, we studied the adaptation of text-to-text models pretrained on natural language for graph-based inputs and analyzed the impact of different task-adaptive pretraining strategies for graph-to-text generation across three graph domains: meaning representations, Wikipedia and scientific knowledge graphs. Further, in Chapter 8 we continuously adjusted these methods to handle complex graph structures, implementing an adapter method that injects graph connectivity into the pretrained model, only training a small fraction of the original model parameters. We found that our technique is more effective when encoding complex graphs, when trained on a limited number of examples, and is more robust to different graph linearizations. Finally, in Chapter 9, we proposed multilingual multi-task training methods to create novel multilingual models capable of decoding into over four different languages from the same structured AMR. The proposed approaches – task-adaptive pretraining, structural adapters and multilingual multi-task structural training – effectively adapt pretrained language models to graph-based inputs leading to state-of-the-art results in different scenarios. Our proposed techniques facilitate the practical use of the promising pretrained language models for downstream text generations that consume structured data.

10.2 Future Work

We end this work by discussing interesting research directions from distinctive perspectives. The focus of this thesis is on better encoding into neural models meaning and knowledge stored in graph-based inputs, with the premise that this structured semantic information is external and must be incorporated into the model for producing text. Recent research has shown significant potential for neural models that store in their parameters implicit knowledge learned through training on extremely large corpora (Petroni et al., 2019; Roberts et al., 2020). This implicit knowledge can be extracted from those pretrained models to fill in masked facts presented in natural text created using knowledge graph triples. Combining external and implicit knowledge in order to better represent the information to be realized in natural language is an important area of study. For example, knowledge graphs are known to be incomplete, and facts extracted from pretrained models can be exploited to complete a partial subgraph. Furthermore, various semantic representations such as Abstract Meaning Representation (AMR) do not fully represent co-references that cross sentence boundaries, and the implicit knowledge encoded in pretrained models can help to capture relations between entities across different sentences.

While studying multilingual text generation for AMR-to-text models in Chapter 9, we witnessed that current multilingual data for structured representations is scarce. Machine translation approaches can be employed for translating knowledge graph triples, meaning representations and the target text. Furthermore, one may think of the use of cross-lingual transfer learning (Liu et al., 2020a; Xue et al., 2021) to generate multilingual text or the extraction of entity mappings from multilingual resources (De Cao et al., 2022) to augment or create graph relations and node/edge attributes. In such cases, a text generation model for multiple languages would be

enhanced with additional relations and entities constructed from multilingual resources. Moreover, it would be particularly interesting to study multilingual models in zero-shot or few-shot graph-to-text settings. In this scenario, natural language data used for training would consist of only a few languages, while the models would be evaluated in unseen languages.

The capacity to scale neural encoder-decoder models to many millions, if not billions, of learned parameters is one of the main causes of their success. While the performance of such large models on diverse text generation tasks is impressive, this capability comes at the cost of increasing latency and energy consumption due to the high computational costs. While larger models continue to be developed, recent studies propose various strategies for making such models smaller and faster with relatively minor performance drawbacks. It will be vital to be able to distill, quantize or minimize model size while maintaining performance (Sanh et al., 2019; Jiao et al., 2020). This is particularly important for text generation models since those approaches employ expensive decoding algorithms (e.g., beam search). In this way, such models can be used in devices with limited computation capacity and storage, such as smartphones and smartwatches, enabling their usage for a wide range of users.

Finally, it will be necessary to conduct a more in-depth study on the limitations of end-to-end neural systems for text generation from structured data. Traditional data-to-text generation systems were inherently grounded and controllable due to a planning phase that was essential in organizing and ordering the data and anchoring the text generation to the plan (Castro Ferreira et al., 2019). Although modern neural generation systems have advanced text generation beyond initial expectations, some of the most desired features, such as controllability and grounding, are still challenging to be implemented (Narayan et al., 2021). In fact, many current models suffer from a severe limitation, generating text that is not factually consistent, that is, the content of the generated text does not meet all the facts of the input structured data. This limitation is especially prominent in out-of-domain scenarios. As we consider the creation of models that truly work for people worldwide in different applications, we are required to develop novel neural generation techniques that function in a controllable, inspectable, and trustworthy way.

Bibliography

Roee Aharoni, Melvin Johnson, and Orhan Firat. 2019. Massively multilingual neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3874–3884, Minneapolis, Minnesota. Association for Computational Linguistics.

Wasi Ahmad, Saikat Chakraborty, Baishakhi Ray, and Kai-Wei Chang. 2020. A transformer-based approach for source code summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4998–5007, Online. Association for Computational Linguistics.

Uri Alon and Eran Yahav. 2021. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv e-prints*.

Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Valerio Basile and Johan Bos. 2011. Towards generating text from discourse representation structures. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 145–150, Nancy, France. Association for Computational Linguistics.

Jasmijn Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1957–1967, Copenhagen, Denmark. Association for Computational Linguistics.

Anja Belz. 2008. Automatic generation of weather forecast texts using compre-

hensive probabilistic generation-space models. *Natural Language Engineering*, 14(4):431–455.

Anja Belz, Michael White, Dominic Espinosa, Eric Kow, Deirdre Hogan, and Amanda Stent. 2011. The first surface realisation shared task: Overview and evaluation results. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 217–226, Nancy, France. Association for Computational Linguistics.

Kalina Bontcheva and Yorick Wilks. 2004. Automatic report generation from ontologies: The miakt approach. In *Natural Language Processing and Information Systems*, pages 324–335, Berlin, Heidelberg. Springer Berlin Heidelberg.

Mihaela Bornea, Ramon Fernandez Astudillo, Tahira Naseem, Nandana Mihindukulasooriya, Ibrahim Abdelaziz, Pavan Kapanipathi, Radu Florian, and Salim Roukos. 2021. Learning to transpile AMR into SPARQL. *arXiv e-prints*.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Ralf Brown and Robert Frederking. 1995. Applying statistical English language modelling to symbolic machine translation. In *Proceedings of the Sixth Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, Katholieke Universiteit, Leuven.

Aoife Cahill and Josef van Genabith. 2006. Robust PCFG-based generation using automatically acquired LFG approximations. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 1033–1040, Sydney, Australia. Association for Computational Linguistics.

Thiago Castro Ferreira, Claire Gardent, Nikolai Ilinykh, Chris van der Lee, Simon Mille, Diego Moussallem, and Anastasia Shimorina. 2020. The 2020 bilingual, bidirectional WebNLG+ shared task: Overview and evaluation results (WebNLG+ 2020). In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 55–76, Dublin, Ireland (Virtual). Association for Computational Linguistics.

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

Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. 2021. Evaluation of text generation: A survey. *arXiv e-prints*.

David L. Chen and Raymond J. Mooney. 2008. Learning to sportscast: A test of grounded language acquisition. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 128–135, New York, NY, USA. Association for Computing Machinery.

Marco Damonte and Shay B. Cohen. 2018. Cross-lingual Abstract Meaning Representation parsing. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1146–1155, New Orleans, Louisiana. Association for Computational Linguistics.

Marco Damonte, Rahul Goel, and Tagyoung Chung. 2019. Practical semantic parsing for spoken language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Industry Papers)*, pages 16–23, Minneapolis, Minnesota. Association for Computational Linguistics.

Nicola De Cao, Ledell Wu, Kashyap Popat, Mikel Artetxe, Naman Goyal, Mikhail Plekhanov, Luke Zettlemoyer, Nicola Cancedda, Sebastian Riedel, and Fabio Petroni. 2022. Multilingual Autoregressive Entity Linking. *Transactions of the Association for Computational Linguistics*, 10:274–290.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Shibhansh Dohare, Vivek Gupta, and Harish Karnick. 2018. Unsupervised semantic abstractive summarization. In *Proceedings of ACL 2018, Student Research Workshop*, pages 74–83, Melbourne, Australia. Association for Computational Linguistics.

Tobias Falke, Leonardo F. R. Ribeiro, Prasetya Ajie Utama, Ido Dagan, and Iryna Gurevych. 2019. Ranking generated summaries by correctness: An interesting but challenging application for natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2214–2220, Florence, Italy. Association for Computational Linguistics.

Angela Fan, Claire Gardent, Chloé Braud, and Antoine Bordes. 2019. Using local knowledge graph construction to scale Seq2Seq models to multi-document inputs. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4186–4196, Hong Kong, China. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini.

2017. The WebNLG challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133, Santiago de Compostela, Spain. Association for Computational Linguistics.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61(1):65–170.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252. PMLR.

Sebastian Gehrmann, Tosin Adewumi, Karmanya Aggarwal, Pawan Sasanka Ammanamanchi, Anuoluwapo Aremu, Antoine Bosselut, Khyathi Raghavi Chandu, Miruna-Adriana Clinciu, Dipanjan Das, Kaustubh Dhole, Wanyu Du, Esin Durmus, Ondřej Dušek, Chris Chinenye Emezue, Varun Gangal, Cristina Garbacea, Tatsunori Hashimoto, Yufang Hou, Yacine Jernite, Harsh Jhamtani, Yangfeng Ji, Shailza Jolly, Mihir Kale, Dhruv Kumar, Faisal Ladhak, Aman Madaan, Mounica Maddela, Khyati Mahajan, Saad Mahamood, Bodhisattwa Prasad Majumder, Pedro Henrique Martins, Angelina McMillan-Major, Simon Mille, Emiel van Miltenburg, Moin Nadeem, Shashi Narayan, Vitaly Nikolaev, Andre Niyongabo Rubungo, Salomey Osei, Ankur Parikh, Laura Perez-Beltrachini, Niranjan Ramesh Rao, Vikas Raunak, Juan Diego Rodriguez, Sashank Santhanam, João Sedoc, Thibault Sellam, Samira Shaikh, Anastasia Shimorina, Marco Antonio Sobrevilla Cabezudo, Hendrik Strobelt, Nishant Subramani, Wei Xu, Diyi Yang, Akhila Yerukola, and Jiawei Zhou. 2021. The GEM benchmark: Natural language generation, its evaluation and metrics. In *Proceedings of the 1st Workshop on Natural Language Generation, Evaluation, and Metrics (GEM 2021)*, pages 96–120, Online. Association for Computational Linguistics.

Dale Gerdemann and Erhard W. Hinrichs. 1990. Functor-driven natural language generation with categorial-unification grammars. In *Proceedings of the 13th Conference on Computational Linguistics - Volume 2*, COLING '90, page 145–150, USA. Association for Computational Linguistics.

Marco Gori, Gabriele Monfardini, and Franco Scarselli. 2005. A new model for learning in graph domains. *Proceedings of the 2005 IEEE International Joint Conference on Neural Networks, 2005.*, 2:729–734 vol. 2.

Valerie Hajdik, Jan Buys, Michael Wayne Goodman, and Emily M. Bender. 2019. Neural text generation from rich semantic representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2259–2266, Minneapolis, Minnesota. Association for Computational Linguistics.

David K. Hammond, Pierre Vandergheynst, and Rémi Gribonval. 2011. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150.

Hardy Hardy and Andreas Vlachos. 2018. Guided neural language generation for abstractive summarization using Abstract Meaning Representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 768–773, Brussels, Belgium. Association for Computational Linguistics.

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

Xiaodong He and Li Deng. 2017. Deep learning for image-to-text generation: A technical overview. *IEEE Signal Processing Magazine*, 34(6):109–116.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

J. J. Hopfield. 1982. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, 79(8):2554–2558.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Alexander Miserlis Hoyle, Ana Marasović, and Noah A. Smith. 2021. Promoting graph awareness in linearized graph-to-text generation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 944–956, Online. Association for Computational Linguistics.

Luyang Huang, Lingfei Wu, and Lu Wang. 2020. Knowledge graph-augmented abstractive summarization with semantic-driven cloze reward. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5094–5107, Online. Association for Computational Linguistics.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Juraj Juraska, Kevin Bowden, and Marilyn Walker. 2019. ViGGO: A video game corpus for data-to-text generation in open-domain conversation. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 164–172, Tokyo, Japan. Association for Computational Linguistics.

Pavan Kapanipathi, Ibrahim Abdelaziz, Srinivas Ravishankar, Salim Roukos, Alexander Gray, Ramón Fernandez Astudillo, Maria Chang, Cristina Cornelio, Saswati Dana, Achille Fokoue, Dinesh Garg, Alfio Gliozzo, Sairam Gurajada, Hima Karanam, Naweed Khan, Dinesh Khandelwal, Young-Suk Lee, Yunyao Li, Francois Luus, Ndivhuwo Makondo, Nandana Mihindukulasooriya, Tahira Naseem, Sumit Neelam, Lucian Popa, Revanth Gangi Reddy, Ryan Riegel, Gaetano Rossiello, Udit Sharma, G P Shrivatsa Bhargav, and Mo Yu. 2021. Leveraging Abstract Meaning Representation for knowledge base question answering. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3884–3894, Online. Association for Computational Linguistics.

David Kauchak and Regina Barzilay. 2006. Paraphrasing for automatic evaluation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 455–462, New York City, USA. Association for Computational Linguistics.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text Generation from Knowledge Graphs with Graph Transformers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2284–2293, Minneapolis, Minnesota. Association for Computational Linguistics.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157, Vancouver, Canada. Association for Computational Linguistics.

Ioannis Konstas and Mirella Lapata. 2013. A global model for concept-to-text generation. *Journal of Artificial Intelligence Research*, 48(1):305–346.

Karen Kukich. 1983. Design of a knowledge-based report generator. In *21st Annual Meeting of the Association for Computational Linguistics*, pages 145–150, Cambridge, Massachusetts, USA. Association for Computational Linguistics.

Anne Lauscher, Olga Majewska, Leonardo F. R. Ribeiro, Iryna Gurevych, Nikolai Rozanov, and Goran Glavaš. 2020. Common sense or world knowledge? investigat-

ing adapter-based knowledge injection into pretrained transformers. In *Proceedings of Deep Learning Inside Out (DeeLIO): The First Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 43–49, Online. Association for Computational Linguistics.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213, Austin, Texas. Association for Computational Linguistics.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature*, 521(7553):436.

Chris van der Lee, Emiel Krahmer, and Sander Wubben. 2017. PASS: A Dutch data-to-text system for soccer, targeted towards specific audiences. In *Proceedings of the 10th International Conference on Natural Language Generation*, INLG'2017, pages 95–104, Santiago de Compostela, Spain. Association for Computational Linguistics.

Fei-Tzin Lee, Chris Kedzie, Nakul Verma, and Kathleen McKeown. 2021. An analysis of document graph construction methods for amr summarization. *arXiv e-prints*.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2015. Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195.

Leo Leppänen, Myriam Munezero, Mark Granroth-Wilding, and Hannu Toivonen. 2017. 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.

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

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

Jungwoo Lim, Dongsuk Oh, Yoonna Jang, Kisu Yang, and Heuiseok Lim. 2020. I know what you asked: Graph path learning using AMR for commonsense reasoning. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2459–2471, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2018. Table-to-text generation by structure-aware seq2seq learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press.

Yang Liu and Mirella Lapata. 2019. 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 (EMNLP-IJCNLP)*, pages 3730–3740, Hong Kong, China. Association for Computational Linguistics.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020a. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020b. Roberta: A robustly optimized bert pretraining approach. *arXiv e-prints*.

Wei Lu and Hwee Tou Ng. 2011. A probabilistic forest-to-string model for language generation from typed lambda calculus expressions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1611–1622, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Iain Macdonald and Advaith Siddharthan. 2016. Summarising news stories for children. In *Proceedings of the 9th International Natural Language Generation conference*, pages 1–10, Edinburgh, UK. Association for Computational Linguistics.

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

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515, Copenhagen, Denmark. Association for Computational Linguistics.

Kathleen R. McKeown. 1982. The text system for natural language generation: An overview. In *20th Annual Meeting of the Association for Computational Linguistics*, pages 113–120, Toronto, Ontario, Canada. Association for Computational Linguistics.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? selective generation using LSTMs with coarse-to-fine alignment. In *Proceedings of the 2016 Conference of the North American Chapter of the Association*

for Computational Linguistics: Human Language Technologies, HLT-NAACL'16, pages 720–730, San Diego, California. Association for Computational Linguistics.

Mohsen Mesgar, Leonardo F. R. Ribeiro, and Iryna Gurevych. 2021. A neural graph-based local coherence model. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2316–2321, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Noelia Migueles-Abraira, Rodrigo Agerri, and Arantza Diaz de Ilarraza. 2018. Annotating Abstract Meaning Representations for Spanish. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Simon Mille, Anja Belz, Bernd Bohnet, Yvette Graham, Emily Pitler, and Leo Wanner. 2018. The first multilingual surface realisation shared task (SR'18): Overview and evaluation results. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 1–12, Melbourne, Australia. Association for Computational Linguistics.

Shashi Narayan, Yao Zhao, Joshua Maynez, Gonçalo Simões, Vitaly Nikolaev, and Ryan McDonald. 2021. Planning with learned entity prompts for abstractive summarization. *Transactions of the Association for Computational Linguistics*, 9:1475–1492.

Franz Josef Och, Christoph Tillmann, and Hermann Ney. 1999. Improved alignment models for statistical machine translation. In *1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. ToTTo: A controlled table-to-text generation dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186, Online. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Vassilis Plachouras, Charese Smiley, Hiroko Bretz, Ola Taylor, Jochen L. Leidner, Dezhao Song, and Frank Schilder. 2016. Interacting with financial data using natural language. In *SIGIR*, pages 1121–1124. ACM.

François Portet, Ehud Reiter, Albert Gatt, Jim Hunter, Somayajulu Sripada, Yvonne Freer, and Cindy Sykes. 2009. Automatic generation of textual summaries from neonatal intensive care data. *Artificial Intelligence*, 173(7–8):789 – 816.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. ProphetNet: Predicting future n-gram for sequence-to-SequencePre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2401–2410, Online. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *arXiv e-prints*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Sudha Rao and Hal Daumé III. 2018. Learning to ask good questions: Ranking clarification questions using neural expected value of perfect information. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2737–2746, Melbourne, Australia. Association for Computational Linguistics.

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press, USA.

Leonardo F. R. Ribeiro, Claire Gardent, and Iryna Gurevych. 2019. Enhancing AMR-to-text generation with dual graph representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3183–3194, Hong Kong, China. Association for Computational Linguistics.

Leonardo F. R. Ribeiro, Jonas Pfeiffer, Yue Zhang, and Iryna Gurevych. 2021a. Smelting gold and silver for improved multilingual AMR-to-Text generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 742–750, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

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

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

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online. Association for Computational Linguistics.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. 2021. AdapterDrop: On the efficiency of adapters in transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7930–7946, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning Representations by Back-propagating Errors. *Nature*, 323(6088):533–536.

Vasile Rus, Brendan Wyse, Paul Piwek, Mihai Lintean, Svetlana Stoyanchev, and Cristian Moldovan. 2011. Question generation shared task and evaluation challenge – status report. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 318–320, Nancy, France. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. In *5th Workshop on Energy Efficient Machine Learning and Cognitive Computing @ NeurIPS 2019*.

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

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

Yuanlong Shao, Stephan Gouws, Denny Britz, Anna Goldie, Brian Strope, and Ray Kurzweil. 2017. Generating high-quality and informative conversation responses with sequence-to-sequence models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2210–2219, Copenhagen, Denmark. Association for Computational Linguistics.

Anastasia Shimorina, Elena Khasanova, and Claire Gardent. 2019. Creating a corpus for Russian data-to-text generation using neural machine translation and post-editing. In *Proceedings of the 7th Workshop on Balto-Slavic Natural Lan-*

guage Processing, pages 44–49, Florence, Italy. Association for Computational Linguistics.

Chang Shu, Yusen Zhang, Xiangyu Dong, Peng Shi, Tao Yu, and Rui Zhang. 2021. Logic-consistency text generation from semantic parses. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4414–4426, Online. Association for Computational Linguistics.

Marco Antonio Sobrevilla Cabezudo and Thiago Pardo. 2019. Towards a general Abstract Meaning Representation corpus for Brazilian Portuguese. In *Proceedings of the 13th Linguistic Annotation Workshop*, pages 236–244, Florence, Italy. Association for Computational Linguistics.

Linfeng Song, Daniel Gildea, Yue Zhang, Zhiguo Wang, and Jinsong Su. 2019. Semantic neural machine translation using AMR. *Transactions of the Association for Computational Linguistics*, 7:19–31.

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

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 4444–4451. AAAI Press.

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

Tao Tao, Xuanhui Wang, Qiaozhu Mei, and ChengXiang Zhai. 2006. Language model information retrieval with document expansion. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 407–414, New York City, USA. Association for Computational Linguistics.

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

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

Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond J. Mooney, Trevor Darrell, and Kate Saenko. 2015. Sequence to sequence - video to text. In *ICCV*, pages 4534–4542. IEEE Computer Society.

Pavlos Vougiouklis, Hady Elsahar, Lucie-Aimée Kaffee, Christophe Gravier, Frédérique Laforest, Jonathon Hare, and Elena Simperl. 2018. Neural wikipedian:

Generating textual summaries from knowledge base triples. *Journal of Web Semantics*, 52-53:1 – 15.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledgebase. *Communications of the Association for Computing Machinery*, 57(10):78–85.

Yao Wan, Zhou Zhao, Min Yang, Guandong Xu, Haochao Ying, Jian Wu, and Philip S. Yu. 2018. *Improving Automatic Source Code Summarization via Deep Reinforcement Learning*, page 397–407. Association for Computing Machinery, New York, NY, USA.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721, Lisbon, Portugal. Association for Computational Linguistics.

Michael White, Rajakrishnan Rajkumar, and Scott Martin. 2007. Towards broad coverage surface realization with CCG. In *Proceedings of the Workshop on Using corpora for natural language generation*, Copenhagen, Denmark.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

David S Wishart, Yannick D Feunang, An C Guo, Elvis J Lo, Ana Marcu, Jason R Grant, Tanvir Sajed, Daniel Johnson, Carin Li, Zinat Sayeeda, Nazanin Assempour, Ithayavani Iynkkaran, Yifeng Liu, Adam Maciejewski, Nicola Gale, Alex Wilson, Lucy Chin, Ryan Cummings, Diana Le, Allison Pon, Craig Knox, and Michael Wilson. 2017. DrugBank 5.0: a major update to the DrugBank database for 2018. *Nucleic Acids Research*, 46(D1):D1074–D1082.

Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I. Wang, Victor Zhong, Bailin Wang, Chengzu Li, Connor Boyle, Ansong Ni, Ziyu Yao, Dragomir Radev, Caiming Xiong, Lingpeng Kong, Rui Zhang, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. 2022. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018a. Representation learning on graphs with jumping knowledge networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5453–5462. PMLR.

Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, and Vadim Sheinin. 2018b. SQL-to-text generation with graph-to-sequence model. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 931–936, Brussels, Belgium. Association for Computational Linguistics.

Wei Xu, Alan Ritter, Bill Dolan, Ralph Grishman, and Colin Cherry. 2012. Paraphrasing for style. In *Proceedings of COLING 2012*, pages 2899–2914, Mumbai, India. The COLING 2012 Organizing Committee.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32, pages 5753–5763. Curran Associates, Inc.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19. AAAI Press.

Chengxiang Zhai and John Lafferty. 2001. Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of the Tenth International Conference on Information and Knowledge Management*, CIKM '01, page 403–410, New York, NY, USA. Association for Computing Machinery.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 11328–11339. PMLR.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 584–594, Copenhagen, Denmark. Association for Computational Linguistics.

Chao Zhao, Marilyn Walker, and Snigdha Chaturvedi. 2020. Bridging the structural gap between encoding and decoding for data-to-text generation. In *Proceedings of*

the 58th Annual Meeting of the Association for Computational Linguistics, pages 2481–2491, Online. Association for Computational Linguistics.

Appendix A

Data Handling

In accordance with DFG's "Principles for the Handling of Research Data",[1] we ensured the long-term preservation of research data and/or experimental software that has been developed as part of this dissertation. We made this data openly accessible when possible. The following software has been made available for the scientific community (see the repositories for licensing details):

- Chapter 4: https://github.com/UKPLab/emnlp2019-dualgraph
- Chapter 5: https://github.com/UKPLab/kg2text
- Chapter 6: https://github.com/mnschmit/graformer
- Chapter 7: https://github.com/UKPLab/plms-graph2text
- Chapter 8: https://github.com/UKPLab/StructAdapt
- Chapter 9: https://github.com/UKPLab/m-AMR2Text

Our trained models are distributed via UKP Lab's public webserver, due to the large size of the data (License: Creative Commons Attribution Share-Alike 4.0):

- Chapter 4: https://public.ukp.informatik.tu-darmstadt.de/ribeiro/emnlp19_dualgraph
- Chapters 5 and 7: https://public.ukp.informatik.tu-darmstadt.de/ribeiro/graph2text
- Chapter 8: https://public.ukp.informatik.tu-darmstadt.de/ribeiro/structadapt
- Chapter 9: https://public.ukp.informatik.tu-darmstadt.de/ribeiro/

[1] https://www.dfg.de/download/pdf/foerderung/grundlagen_dfg_foerderung/forschungsdaten/leitlinien_forschungsdaten.pdf

graph2text/mt5_base_silveramr_silversent.tar.gz

All publications related to this thesis are publicly available on the ACL Anthology (aclweb.org/anthology/):

- Chapter 4: https://aclanthology.org/D19-1314/
- Chapter 5: https://aclanthology.org/2020.tacl-1.38/
- Chapter 6: https://aclanthology.org/2021.textgraphs-1.2/
- Chapter 7: https://aclanthology.org/2021.nlp4convai-1.20/
- Chapter 8: https://aclanthology.org/2021.emnlp-main.351/
- Chapter 9: https://aclanthology.org/2021.emnlp-main.57/

Moreover, all research results of the aforementioned publications are documented in the present thesis, which is archived by the Universitäts- und Landesbibliothek Darmstadt.