Investigating the Effect of Relative Positional Embeddings on AMR-to-Text Generation with Structural Adapters

Sebastien Montella*
Orange Innovation / Lannion, France
Aix-Marseille Univ. CNRS, LIS / Marseille, France
sebastien.montella@huawei.com

Alexis Nasr
AMU/CNRS/LIS, Marseille, France
alexis.nasr@lis-lab.fr

Johannes Heinecke
Orange Innovation / Lannion, France
johannes.heinecke@orange.com

Frederic Bechet
AMU/CNRS/LIS, Marseille, France
frederic.bechet@lis-lab.fr

Lina M. Rojas-Barahona
Orange Innovation / Lannion, France
linamaria.rojasbarahona@orange.com

Abstract

Text generation from Abstract Meaning Representation (AMR) has substantially benefited from the popularized Pretrained Language Models (PLMs). Myriad approaches have linearized the input graph as a sequence of tokens to fit the PLM tokenization requirements. Nevertheless, this transformation jeopardizes the structural integrity of the graph and is therefore detrimental to its resulting representation. To overcome this issue, Ribeiro et al. (2021b) have recently proposed StructAdapt, a structure-aware adapter which injects the input graph connectivity within PLMs using Graph Neural Networks (GNNs). In this paper, we investigate the influence of Relative Position Embeddings (RPE) on AMR-to-Text, and, in parallel, we examine the robustness of StructAdapt. Through ablation studies, graph attack and link prediction, we reveal that RPE might be partially encoding input graphs. We suggest further research regarding the role of RPE will provide valuable insights for Graph-to-Text generation.

1 Introduction

Earliest works on AMR-to-Text generation were mostly based on statistical methods. A common practice was to convert AMR-to-Text task into an already studied problems such as Tree-to-Text (Flanigan et al., 2016; Lampouras and Vlachos, 2017), aligned text-to-text (Pourdamghani et al., 2016), Travel Sales Problems (Song et al., 2016) or Grammatical Framework (Ranta, 2011). Recently, most methods are neural-centered with an encoder-decoder architecture (Sutskever et al., 2014) as a backbone (Konstas et al., 2017; Takase et al., 2016; Cao and Clark, 2019). Unfortunately, this architecture coerces the AMR to be linearized as a sequence of tokens. This ends up in structural information loss. To tackle this issue, several strategies have attempted to integrate structure using message propagation (Song et al., 2018; Guo et al., 2019; Damonte and Cohen, 2019; Ribeiro et al., 2019; Zhang et al., 2020b; Zhao et al., 2020). A limitation of those is the absence of pretraining, as demonstrated by Ribeiro et al. (2021a). To this end, Ribeiro et al. (2021b) introduced StructAdapt for lightweight AMR-to-Text with structural adapters. As linearization and tokenization of the input graph are mandatory steps for PLMs, StructAdapt first defines a new graph where nodes are the resulting subwords from the tokenization. As a result, adapter can henceforth include GNN layers operating on the subsequent graph while leveraging pretrained representations.

However, although studies have been made to probe position embeddings (Wang and Chen, 2020; Wang et al., 2021; Dufter et al., 2022), their role on graph encoding has remained unanswered. In this

*Work done while at Orange Innovation (Lannion, France). Now at Huawei Edinburgh Research Centre (United Kingdom).

paper, we are particularly interested in measuring the saliency of RPE with StructAdapt for AMR-to-Text generation. Our novelty is not in proposing a new method to encode graphs such as (Schmitt et al., 2021) but rather in revealing the interesting behaviours of RPE along with StructAdapt.

2 StructAdapt: A Structural Adapter

A major issue in AMR-to-Text, and more generally Graph-to-Text with Transformers (Vaswani et al., 2017), is the linearization of the input structure. The linearization of the graph returns a sequence of node and edge labels according to a certain traversal of the graph. Nonetheless, adjacent nodes in the graph may be at multiple positions away from one another in the final serialization. To counteract this, Ribeiro et al. (2021b) introduced StructAdapt, a structure-aware (encoder) adapter. It solves the problem of segmented nodes labels by reconstructing a new graph from the resulting subwords. More specifically, the relations are primary reified as new nodes in the AMR graph. Furthermore, labels of those reified relations will be added in the vocabulary as new tokens and therefore will not be decomposed into subwords. However, the labels of the original nodes can still be chunked. To deal with this, each subword node will be connected independently to the reified relation of the initial (non-chunked) node. An example is outlined in Figure 1. As a consequence, the vanilla Adapter can now integrate any GNN-based neural network which operates on the new constructed graph (Figure 1), where nodes are the input tokens. Concretely, StructAdapt replaces the first stacked MLP of vanilla adapter with a GNN-based model as shown in Figure 2. For AMR-to-Text, only the encoder is equipped with StructAdapt in order to encode AMR structure. The decoder layers adopt vanilla adapters. In our study, we consider three different GNN-based models which are Graph Convolutional Network (GCN) (Kipf and Welling, 2017), Graph Attention network (GAT) (Veličković et al., 2017) and Relational Graph Convolutional Network (RGCN) (Schlichtkrull et al., 2018). GCN computes a representation for each node a which is a (normalized) aggregation function of representation of its neighbor nodes noted $\mathcal{N}(a)$. GAT is akin to GCN but differs in that aggregation of neighbors embeddings are weighted using an attention mechanism. Unlike GAT and GCN, RGCN further captures the type of the relation between

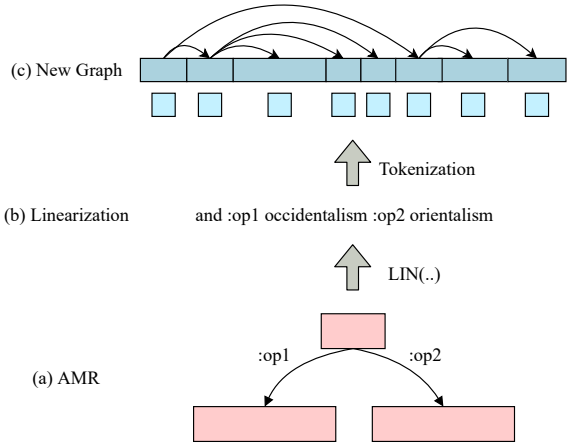

Figure 1: Examples of AMR tokenization for the sentence *"Occidentalism and Orientialism."*. The resulting input graph in (c) contains 8 nodes.

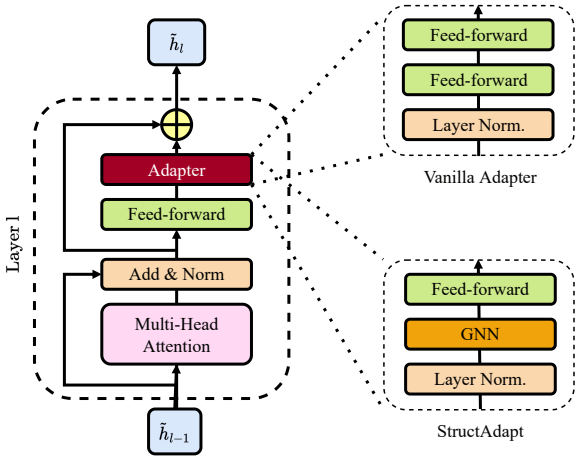

Figure 2: Vanilla Adapter vs StructAdapt

two nodes. In our case, as AMR relations are reified and stand for new nodes of the graphs, our new relations can either be of *direct* ($a \xrightarrow{d} b$) or *reverse* ($a \xleftarrow{r} b$) connections type as in (Ribeiro et al., 2021b). The details of the representations computation for each model can be found in Appendix A. The returned nodes embeddings will then be given as input features to the following MLP.

3 Relative Position Embeddings

Instead of adding Absolute Position Embeddings (APE) directly to the token embedding as in standard Transformer model, some models such as T5 make use of relative position embeddings inspired from Shaw et al. (2018). As an alternative to APE, RPE offer interesting features. A noteworthy limitation of APE is the need to set a limit of available positions. Therefore, long sequences may have

Adapter		BLEU ↑	METEOR ↑	Chrf++ ↑	TER ↓	BERTScore ↑	$\mathcal{M}$ ↑	$\mathcal{F}$ ↑
MLP	**w/ RPE**	**41.6**±0.3	**56.5**±0.3	**70.6**±0.2	**45.9**±0.5	**95.6**±0.0	**84.2**±0.1	**78.0**±0.5
	w/o RPE	16.3±0.5	38.5±0.8	49.9±1.1	85.5±0.9	91.8±0.2	81.9±1.1	76.2±0.6
GCN	**w/ RPE**	**42.6**±0.8	**56.7**±0.4	**71.0**±0.5	**44.8**±0.7	**95.7**±0.1	**84.6**±0.3	**79.0**±0.6
	w/o RPE	34.4±0.8	52.0±0.7	64.8±0.6	55.8±1.2	94.6±0.1	79.2±0.6	75.2±1.0
GAT	**w/ RPE**	**42.8**±0.1	**57.0**±0.1	**71.1**±0.0	**44.3**±0.3	**95.8**±0.0	**84.8**±0.1	**78.5**±0.8
	w/o RPE	34.8±1.1	52.3±0.7	64.8±0.7	54.9±1.4	94.6±0.2	79.6±0.8	75.6±0.4
RGCN	**w/ RPE**	**44.7**±0.6	**58.2**±0.3	**72.5**±0.3	**42.6**±0.4	**96.0**±0.0	**85.5**±0.2	**79.6**±0.7
	w/o RPE	39.9±0.8	55.7±0.5	68.9±0.8	48.8±1.3	95.3±0.1	83.1±0.4	78.0±0.6

Table 1: Comparing impact of Relative Positional Embeddings (RPE) on generation. We report mean performances (±s.d.) over 3 seeds.

Adapter	Meaning Preservation	Linguistic Correctness
MLP w/ RPE	**4.8**±1.2	**5.5**±0.9
MLP w/o RPE	3.5±1.5	5.2±1.2
GCN w/ RPE	**5.0**±1.2	**5.6**±0.8
GCN w/o RPE	4.7±1.3	5.5±1.0
RGCN w/ RPE	**5.2**±1.1	**5.6**±0.8
RGCN w/o RPE	4.7±1.3	5.4±1.0

Table 2: Human Evaluation. Mean scores (±s.d.)

to be segmented. Furthermore, APE are directly added to the token representation leading to information inconsistency, i.e. position versus semantic information. To this extent, Shaw et al. (2018) came up with relative position encodings which are supplied to the self-attention mechanism by simply adding a scalar to the logits encoding the supposed relation between a current token i and a token j.

4 Experiments

Throughout our experiments, we make use of the LDC2020T02 dataset (AMR 3.0 release)[1] and use the T5_{base} model which employs RPE. The training and evaluation details can be found in Appendix B and C, respectively.

4.1 Exploring the Salience of RPE

In this section, we investigate the influence of RPE on the generation quality using structural adapter. RPE are computed in each Transformer head. Position information is then forwarded to the adapter module on top (Figure 2). However, since connections between input nodes (i.e. tokens) are already given to structural adapters in encoder, it is legitimate to question the necessity for RPE on the encoder part but also how would the generation quality vary without such information. Hence, we propose to remove the RPE from the encoder heads to gauge their salience to structure encoding for downstream language generation. Since decoder is not encoding any graph structure, we leave RPE in decoder untouched. For better readability, MLP, GCN, GAT, and RGCN respectively denote: the vanilla adapter, StructAdapt with a GCN layer, StructAdapt with a GAT layer and StructAdapt with a RGCN layer. MLP-based adapter with RPE is our baseline. Results are given in Table 1. A human evaluation is also provided for some encoder adapters in Table 2. First, it is apparent that using RPE systematically yields better generation performances. For the vanilla adapter (i.e. our baseline), we note a 25.3% absolute drop in BLEU when removing RPE. This can also be seen on human evaluation. More than one point is lost toward meaning preservation. The downturn for linguistic correctness is less important since T5 is pretrained and thus rarely prone to syntax errors. Such a result is not surprising for MLP-based adapter since it solely relies on RPE to differentiate tokens at different positions in the linearized AMR. However, a striking observation is that getting rid of RPE for GNN-based adapters leads to lower performances than our baseline. Indeed, when removing RPE when using structural adapter, we would have expected GNN-based approaches to be as competitive as a MLP-based adapter with RPE. We report a relative drop of 12.5 points in BLEU from the baseline. The same conclusion can be drawn from Table 2. This indicates that RPE are capturing relevant information for final generation. To further assess the impact and the role of RPE, we conduct a *graph attack* experiment. Instead of conveying the correct adjacency matrix, we propose to corrupt connectivity information. We randomly generate an adjacency matrix such that generated matrix does not contain any actual connection. We suppose that without RPE, structure-aware adapter will lead to a significant decrease in generated text due to the absence of information about the graph nor the position of nodes in the input sequence. We are especially interested to measure to which extent RPE might be able to take over the encoding of the graph for generation.

[1] https://catalog.ldc.upenn.edu/LDC2020T02

Adapter		BLEU ↑	METEOR ↑	Chrf++ ↑	TER ↓	BERTScore ↑	$\mathcal{M}$ ↑	$\mathcal{F}$ ↑
GCN	**w/ RPE**	**35.1**±0.5	**52.0**±0.4	**65.7**±0.5	**53.6**±0.9	**94.8**±0.1	**80.5**±0.4	74.1±1.2
	w/o RPE	13.9±0.5	35.2±0.6	46.7±0.5	86.5±2.7	91.1±0.1	77.1±14.4	**78.0**±2.6
GAT	**w/ RPE**	**36.8**±0.6	**52.3**±0.5	**67.0**±0.5	**51.1**±0.9	**95.0**±0.1	**81.6**±0.5	75.7±0.3
	w/o RPE	13.1±2.8	34.1±3.6	45.2±4.2	89.1±0.6	90.7±0.1	68.9±14.1	**77.4**±1.9
RGCN	**w/ RPE**	**38.0**±0.9	**54.1**±0.6	**67.6**±0.6	**49.3**±0.8	**95.2**±0.1	**81.9**±0.7	**75.9**±0.8
	w/o RPE	11.3±1.3	30.1±1.1	41.6±1.0	87.2±1.3	90.0±0.2	66.8±16.3	75.5±5.2

Table 3: *Graph Attack* - We corrupt the graph connectivity information given to the structural adapters in encoder. We report mean performances (±s.d.) over 3 seeds.

Adapter	Meaning Preservation	Linguistic Correctness
GCN w/ RPE	5.0±1.2	5.6±0.8
RGCN w/ RPE	5.2±1.1	5.6±0.8

Table 4: *Graph Attack* - Human Evaluation. Mean scores (±s.d.)

Results are shown in Table 3. Human evaluation for *attacked* GCN and RGCN adapters is given in Table 4. As hypothesized, providing erroneous connectivity without any position embeddings makes structural adapters no longer compelling. We observe that StructAdapt with RGCN is significantly more affected compared to GAT and GCN based adapters. Since RGCN adds direction information for each edge (*direct* and *reverse*), we conjecture that RGCN is much more bewildered. Interestingly, using RPE with corrupted graph (Table 3) leads to similar performance than using graph information without RPE (Table 1). This strongly demonstrates the usefulness of RPE to carry out the generation. We additionally provide a *position attack* experiment in Appendix E where RPE are shuffled randomly. Accordingly, we can further identify the saliency of RPE despite the available GNN. This raises the question of RPE encoding the input graph.

4.2 Can the Graphs Be Reconstructed ?

As shown in Section 4.1, RPE seem to be as competitive as applying GNNs alone. If claiming that RPE also encode graphs is tempting, no strong evidence has been revealed. Indeed, better generation quality is not necessarily a consequence of better graph encoding. Therefore, we probe whether graphs can indeed be reconstructed from the learned hidden representations. To do so, we train a logistic regression, i.e. our probe, to perform link prediction as a binary classification. More specifically, given two nodes representations at a given layer l, our probe returns the probability that nodes are connected. To train our logistic regression model, we sample k positive connections, i.e. two connected nodes, and k negative connections, i.e. two non-connected nodes, for each sample of training and test sets.[2] For our experiment, we choose $k = 2$ which leads to 109,490 and 3,770 samples for each class for training and testing respectively. We plot results in Figure 3. Firstly, we observe very high accuracy for the vanilla adapter without RPE. Accuracies over 60% are easily reached while no structure encoding nor positions information are supplied. This might be a side effect of our probe training. Nevertheless, this gives a lower bound for our experiment. We can see that adding RPE increases the link prediction performance for MLP, GCN and GAT-based adapters. We observe a constant gap of about 3.5% on average. However, we remark that RGCN is able to reconstruct edges on its own with best accuracy. We report a maximum accuracy about 76% while other models are not reaching 73%. This strengthens the idea that giving information of reverse connection may add robustness to graph encoding as shown in (Beck et al., 2018). Generally, we observe that the deeper the representation, the better the link prediction. We notice however that after the 10^{th} layer, significant drops in link prediction arise regardless

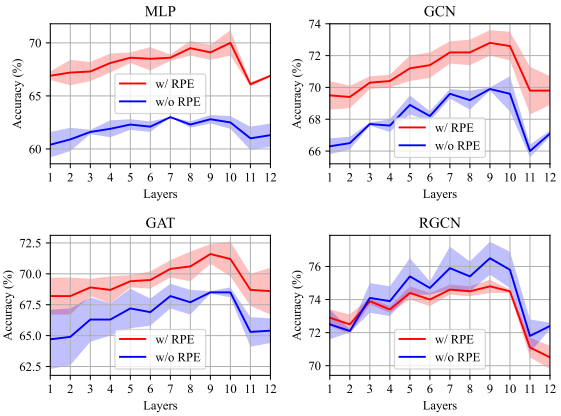

Figure 3: *Probing* - Link Prediction Results using hidden representations $\tilde{h}_i$ at different layers i.

[2] If k samples cannot be extracted for both positive and negative classes, the sample is discarded.

of adapter type. We assume representations should lose some information about the structure to perform language generation. This may indicate that encoded representation for Graph-to-Text is not just graph-centered. Although counter-intuitive, encoder representations given to the decoder part may not have to encode input graph efficiently in order to verbalize it. We leave this research question for future work. We further provide an analysis on self-attention matrices in Appendix F.

5 Conclusion

In this paper, we have explored the effect of relative position embeddings on AMR-to-Text generation using structural adapters. We have shown that the generation process could be enabled by relative position embeddings when structure is erroneous or missing. In addition, we have demonstrated the capacity of those representations to encode the input graph to some extent. We have further revealed interesting robustness of RGCN model in graph reconstruction ability. For future work, we believe further experiments on other pretrained models and Graph-to-Text tasks may shed more light on the role of position embeddings.

Limitations

A limitation of our study is that we focus on the T5 model only. Since adapters are additional modules to add, it is required to manually implement and directly modify the original code of the pretrained model which is not easily scalable. In addition, we only evaluate on the LDC2020T02 dataset which is the cleanest AMR dataset available.

Acknowledgements

We would like to thank annotators and reviewers for taking the time and effort necessary to share our contribution. This work was partially funded by the ANR Cifre conventions N°2020/0400 and Orange Innovation Research.

References

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.

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. 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.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752, Sofia, Bulgaria. Association for Computational Linguistics.

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.

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.

Philipp Dufter, Martin Schmitt, and Hinrich Schütze. 2022. Position information in transformers: An overview. *Computational Linguistics*, 48(3):733–763.

Cédric Fayet, Alexis Blond, Grégoire Coulombel, Claude Simon, Damien Lolive, Gwénolé Lecorvé, Jonathan Chevelu, and Sébastien Le Maguer. 2020. FlexEval, création de sites web légers pour des campagnes de tests perceptifs multimédias (FlexEval, creation of light websites for multimedia perceptual test campaigns). In *Actes de la 6e conférence conjointe Journées d'Études sur la Parole (JEP, 33e édition), Traitement Automatique des Langues Naturelles (TALN, 27e édition), Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RÉCITAL, 22e édition). Volume 4 : Démonstrations et résumés d'articles internationaux*, pages 22–25, Nancy, France. ATALA et AFCP.

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.

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.

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.

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.

Gerasimos Lampouras and Andreas Vlachos. 2017. Sheffield at SemEval-2017 task 9: Transition-based language generation from AMR. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 586–591, Vancouver, Canada. Association for Computational Linguistics.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan. 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.

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.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. 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. https://openai.com/blog/better-language-models/.

Aarne Ranta. 2011. *Grammatical Framework - Programming with Multilingual Grammars*. CSLI Studies in Computational Linguistics. Cambridge University Press.

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

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

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

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web*, pages 593–607, Cham. Springer International Publishing.

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.

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.

Linfeng Song, Yue Zhang, Xiaochang Peng, Zhiguo Wang, and Daniel Gildea. 2016. AMR-to-text generation as a traveling salesman problem. In *Proceedings*

of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2084–2089, Austin, Texas. 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.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3104–3112, Cambridge, MA, USA. MIT Press.

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural headline generation on Abstract Meaning Representation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1054–1059, Austin, Texas. 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 *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2017. Graph attention networks. *6th International Conference on Learning Representations*.

Benyou Wang, Lifeng Shang, Christina Lioma, Xin Jiang, Hao Yang, Qun Liu, and Jakob Grue Simonsen. 2021. On position embeddings in {bert}. In *International Conference on Learning Representations*.

Yu-An Wang and Yun-Nung Chen. 2020. What do position embeddings learn? an empirical study of pre-trained language model positional encoding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6840–6849, Online. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi 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 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*, pages 38–45, Online. Association for Computational Linguistics.

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*. OpenReview.net.

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.

Yanbin Zhao, Lu Chen, Zhi Chen, Ruisheng Cao, Su Zhu, and Kai Yu. 2020. 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.

A Graph Neural Networks

A.1 Graph Convolutional Network (GCN)

To compute the node representation $g_u^l \in \mathbb{R}^{d_g}$ of the node u at layer l, GCN computes an aggregation of neighbors nodes embeddings as:

$$g_u^l = \sigma\left(\sum_{v \in \mathcal{N}(u)} \frac{1}{\deg(u) \times \deg(v)} W^l g_v^l\right) \quad (1)$$

with σ an activation function and $\deg(x)$ the degree of the node x.

A.2 Graph Attention Network (GAT)

GAT applies an attention mechanism to determine importance of neighboors nodes regarding current node u. We have:

$$\begin{aligned} g_u^l &= \sigma\left(\sum_{v \in \mathcal{N}_(u)} \text{softmax}(e_{u,v}) W^l g_v^l\right) \\ e_{u,v} &= \text{LeakyReLu}\left(a_{u,v} \Big\| \overset{v}{\underset{i=u}{} } g_i^l\right) \end{aligned} \quad (2)$$

with both σ and LeakyReLu non-linear activation functions and $a_{u,v} \in \mathbb{R}^{2 \times d_g}$ a learnable vector.

A.3 Relational Graph Convolutional Network (RGCN)

RGCN takes into consideration the nature of the relation $r \in \mathcal{R}$ between nodes u and v. It performs

Adapter	Runtime (in hours)
MLP	21.5
GCN	15.6
GAT	22.0
RGCN	16.2

Table 5: *Runtime* - Runs are executed on a single NVIDIA GeForce RTX 3090 GPU.

convolution as the following:

$$g_u^l = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{v\in\mathcal{N}_r(u)} c_{u,r} W_r^l g_v^l\right) \quad (3)$$

with $\mathcal{N}_r(u)$ the direct neighbors of node u under the relation r, $c_{u,r}$ a normalization term and $W_r^l \in \mathbb{R}^{d_g \times d_g}$ a learnable relation-dependent parameter.

B Training Setting

We used the version 3.3.1 of the HuggingFace's Transformers library (Wolf et al., 2020). We use Adam optimizer with a linearly decreasing learning rate, without warm-up. The initial learning rate is set to 1×10^{-4}. Batch size is set to 8. For decoding, a beam search of 5 is selected. A maximum length of 384 is used in case the *end-of-sentence* token is not encountered. We didn't use gradient clipping nor label smoothing. For GNNs, we make use of the version 1.7.2 of the PyTorch Geometric library (Fey and Lenssen, 2019). We only use a single layer for our GNNs networks, similarly to the vanilla adatper. For GAT, a single attention head was applied. The bottleneck dimension is set to 256 for all of our adapters. This is equivalent to about 4% only of the whole T5$_{base}$ model's parameters to train. The training time for MLP, GCN, GAT and RGCN adapters are given in Table 5. Note that the total runtime depends on the convergence as we are using early stopping.

C Automatic Evaluation

For evaluation, we consider multiple automated metrics. We employ popular n-gram-based metrics which are SacreBLEU (Post, 2018), METEOR (Banerjee and Lavie, 2005), chrf++ (Popović, 2017), TER (Och, 2003) and the semantic-based BertScore metric (Zhang et al., 2020a). In addition, we also report both $\mathcal{M}$ and $\mathcal{F}$ pillars of the decomposable $\mathcal{MF}$ score recently proposed by Opitz and Frank (2021). The meaning preservation, noted $\mathcal{M}$, measures how close the AMR of the generated sentence is to the reference sentence. To do so, both the generated sentence and target sentence are parsed with a pretrained Text-to-AMR model from Cai and Lam (2020). Then, their AMRs are compared using graph-based similarity measures such as Smatch (Cai and Knight, 2013). In contrast, the grammatical form, noted $\mathcal{F}$, measures the linguistic quality of the generated text using state-of-the-art language model (Radford et al., 2019). We also report human evaluation in Section 4.1 to accurately assess the quality of the generated outputs. We ask annotators to evaluate the meaning preservation and linguistic correctness of our generated outputs compared to the references. Details of human evaluation are given in Appendix D.

D Human Evaluation

Since handcrafted annotation is extremely costly, we limited the number of samples and models to assess. We randomly selected 30 test samples per model to evaluate manually. We further limited evaluation to 8 configurations of adapters. This led us with 240 samples to score by multiple annotators. We asked 22 annotators to answer to these following questions with a rate on a 1-6 Likert scale, with 6 the best score:

- Is the generated sentence semantically close to the reference ?
- Is the generated sentence grammatically correct ?

Each annotator was given 50 samples to annotate. We adopted FlexEval (Fayet et al., 2020) to implement a flexible evaluation environment. This allows a streaming evaluation were annotators can stop at any moment and come back to the evaluation later, at one's will. Furthermore, different samples from different models are given to annotators in an balanced manner such that each sample of each model is exposed to at least 2 annotators. Note that one annotator is given some samples from multiple configuration. This avoids evaluation bias toward a single annotator. In our case, each sample from each configuration has been annotated by 4 distinct annotators.

E Position Attack

For a given graph, we also propose to corrupt RPE with a random shuffle. Since the model may learn the generated order through training, we *systematically* shuffle RPE for each epoch for all graphs. Results are given in Table 6.

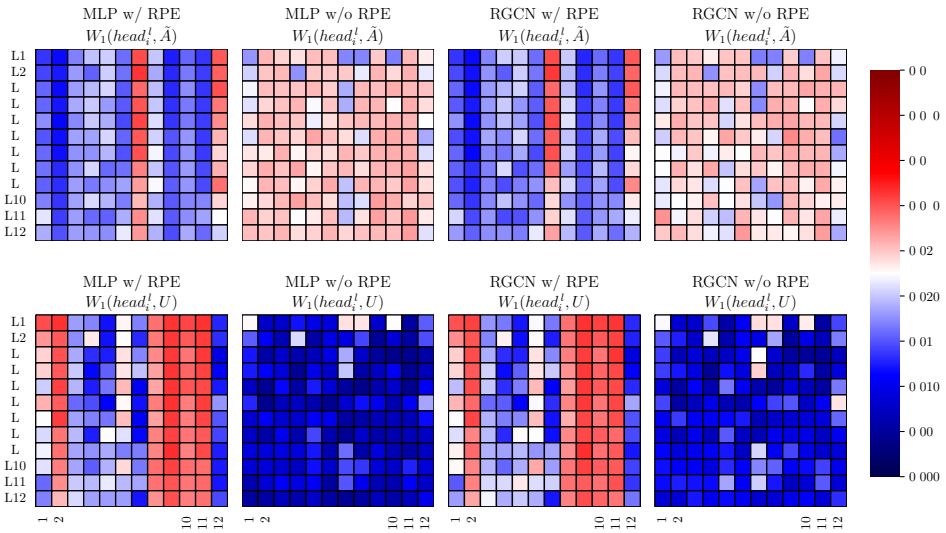

Figure 4: 1-Wasserstein distances of attention distributions for each head (columns) at each layer (rows) from both normalized adjacency matrices (top) and uniform distribution (down). The bluer, the lower the distance.

Adapter	BLEU
MLP	19.0±0.6
GCN	28.1±0.6
GAT	28.9±2.3
RGCN	**35.4**±1.2

Table 6: *Position Attack* - Relative Position Embeddings are shuffled.

F Analysis of Attention Matrices

The self-attention matrix gives information about how much weight should be assigned to other tokens (i.e. nodes). Thus, attention matrix can be seen as a pseudo-adjacency matrix. We further propose to inspect the similarity between the attention matrix of each head at each layer with the adjacency matrix of the input graph. Since the input is considered as a sequence of nodes, the shape of the attention matrix and adjacency matrix are equal. However, a limitation of the similarity comparison lies in the respective type of those mathematical objects. Attention matrices are probabilities whereas adjacency matrices are not. To deal with this issue, we convert the adjacency matrix A^s of a sample s to a normalized form $\tilde{A}^s$ such that each row $\tilde{A}^s_n$ is normalized and sums up to one, as described in Eq. 4.

$$A^s = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \quad \tilde{A}^s = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

This enables the rows of both attention and adjacency matrices to be probabilities distributions over input nodes. We can therefore measure the distance between those distributions to find out whether attention matrices are somehow close to the (normalized) adjacency matrix, and thus encoding graph connections. To do so, we compute the 1-Wasserstein distance W_1 [3] between the attention distribution of each token n and its corresponding normalized n^{th} row of $\tilde{A}$. We average distances over tokens, and then over samples (Eq. 5). We note $\tilde{A}$, S, and N_s the adjacency matrices, the total number of AMR samples and the length of sequence s, respectively.

$$W_1(\text{head}_i^l, \tilde{A}) = \frac{1}{S}\sum_{s=1}^{S}\frac{1}{N_s}\sum_{n=1}^{N_s} W_1(\text{head}_{i,n}^{l,s}, \tilde{A}_n^s) \tag{5}$$

For fair comparison, we also provide the distance between attention distribution head_i^l and the uniform distribution U which assigns a probability of $\frac{1}{N}$ to each token for a graph with N nodes. We plot distances as heatmaps on Figure 4 where each square indicates either $W_1(\text{head}_i^l, \tilde{A})$ or $W_1(\text{head}_i^l, U)$. When using RPE, we can see that, for both MLP and RGCN-based adapters, the distribution of the attention scores are close to the normalized adjacency matrices of our graphs. Meanwhile, when removing them, we can witness

[3] Equivalent to earth mover's distance.

that attention scores are much closer to the uniform distribution and thus smoother. This is in line with our previous results suggesting that RPE might partially encode graphs. Since RPE are shared across layers, we can easily detect similar distances across layers for same heads (visible vertical lines). We also find similar behavior we have previously noted where last layers tend to lose graph information as attention scores slightly move away from the normalized adjacency matrix distribution.