Learning Dual Graph Representations for AMR-to-Text Generation

Leonardo F. R. Ribeiro
Research Training Group AIPHES
and UKP Lab
Technische Universität Darmstadt
ribeiro@aiphes.tu-darmstadt.de

Claire Gardent
CNRS/LORIA
Nancy, France
claire.gardent@loria.fr

Iryna Gurevych
Research Training Group AIPHES
and UKP Lab
Technische Universität Darmstadt
gurevych@ukp.informatik.tu-darmstadt.de

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 new model that encodes the different but complementary perspectives of structural information contained in the AMR graph. Our architecture learns parallel and complementary representations of nodes using state-of-the-art graph encoders. Experimental results demonstrate that the dual graph representation leads to improvements in AMR-to-Text generation, achieving 24.32 BLEU on LDC2015E86 dataset, outperforming recent models and 27.87 BLEU on LDC2017E10 dataset, outperforming state of the art by 3.33 points.

KEYWORDS

AMR, Text Generation, Graph Neural Networks

1 INTRODUCTION

Abstract Meaning Representation (AMR; Banarescu et al. [1]) 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, it is potentially beneficial in many semantic related NLP tasks, including text summarization [18] and machine translation [27].

The purpose of AMR-to-Text generation is to produce a text which verbalises the meaning encoded by the input AMR graph. This is a challenging task as capturing complex structural information stored in graph-based data is not trivial. Recently, Graph Neural Networks (GNNs) have emerged as a powerful class of methods for learning richer graph representation [4, 32]. Previous work have applied GNNs to the task of AMR-to-Text generation [2, 8, 28] directly encoding the graphs. However, they rely on anonymization thereby losing more precise semantic meaning or do not well exploit the graph structure information.

In this paper, we explore the usefulness of encoding different views of the AMR graph. Pre-neural work on Data-to-Text generation focus on modeling different views of the input. Carroll and Oepen [6] propose a bottom-up surface realiser which uses a chart-based generation strategy, whereas Narayan and Gardent [21] present a hybrid bottom-up and top-down model for text generation with lexicalist grammars. Inspired by these pre-neural approaches, we create top-down and bottom-up graph perspectives and employ graph encoders to learn their graph contextualized representation. Therefore, for each concept or relation, each graph encoder specializes in learning a different representation (in an incoming or outgoing perspective). Recent work [2, 8, 15, 19, 28] aggregate all the immediate neighborhood information of a node at the same time. To alleviate this issue, we develop parallel and complementary representations of the AMR graph. Thereby, we ease the burden on the neural model in representing every relation at once. Moreover, it is not required to add positional information [2], since the graphs do not become essentially undirected. Indeed, Moryossef et al. [20] show that high-quality structured inputs enhance the text generation process.

Our proposed models learn richer node representations employing graph encoders based on different classes of GNNs[1]. Experiments on two AMR datasets demonstrate the effectiveness of our approach, leading to improvements that outperform state-of-the-art models in AMR-to-Text generation, including the one that leverages additional syntactic information [5].

2 RELATED WORK

Early work on AMR-to-Text generation employs statistical methods [10, 24] and applies linearization of the graph by means of a depth-first traversal. Recent approaches have exhibited success using encoder-decoder neural architectures. Konstas et al. [16] achieve promising results on this task. However, they strongly rely on the addition of silver training data and anonymization of named entities – our approach, in contrast, allows to omit both of these steps.

A graph-to-sequence model was first introduced by Xu et al. [33] that use a inductive node embedding approach based on the GraphSAGE [12]. Marcheggiani and Perez-Beltrachini [19] show that explicitly encoding the structure of the graph is beneficial with respect to sequential encoding, evaluating their model on two tasks, WebNLG [11] and SR11Deep [3].

Song et al. [28] and Beck et al. [2] apply recurrent networks in their models to learn node embeddings for AMR graphs, in order to generate sentences. In particular, Song et al. [28] use a graph LSTM as the graph encoder, whereas Beck et al. [2] apply a gated recurrent unit (GRU). We go a step further in that direction and develop parallel encodings of graphs which are able to highlight different graph properties. Koncel-Kedziorski et al. [15] propose an attention model for graph encoding based on Graph Attention Networks (GAT) [30], that generates sentences from knowledge graphs.

[1] Our code is available at https://github.com/UKPLab/dlg19-dualgraph2text

Cao and Clark [5] 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). Our work is related to Damonte and Cohen [8] who show that GCNs alone cannot accomplish good performance for AMR-to-Text generation. They use stacking of GCN and LSTM layers to improve the model capacity. However, our model is substantially different: (i) we learn dual representation capturing top-down and bottom-up adjuvant views of the graph, (ii) employ more effective graph encoders (with different neighborhood aggregations) than GCN and (iii) apply copy and coverage mechanisms and do not utilize entity anonymization [16].

3 MODEL IMPLEMENTATION

3.1 Graph preparation

Let $G = (V, E, R)$ denote a rooted 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. $n = |V|$ and $m = |E|$ denote the number of nodes and edges, respectively. We convert each AMR graph into an unlabeled 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 [2], turns original edges into nodes creating an unlabeled graph. The new graph allows us directly representing the relationship between nodes using embeddings.

Finally, we create a graph $G_r = (V_t, E_r)$, where each directed edge $e_k = (v_i, v_j) \in E_t$ becomes $e_k = (v_j, v_i) \in E_r$, that is, we reverse the direction of original edges. Note that G_t and G_r are top-down and bottom-up perspectives of the original graph, respectively.

3.2 Model

We represent each node with a node embedding $\mathbf{n}_i \in \mathbb{R}^d$, generated from the word contained in the node. In order to explicitly encode structural information, our encoder starts with two graph encoders, denoted by GE_t and GE_r, that compute representations for nodes in G_t and G_r, respectively. The goal of GEs is to capture the richer structural representation of each vertex considering the graph structure, that is, each GE focuses on the structure of its particular graph, G_t or G_r. Note that G_t and G_r capture top-down and bottom-up perspectives of the graph structure. Moreover, each graph encoder is forced to focus on its own perspective, learning node representations based on its specific view of the graph. In particular, for each node $v_i \in V_t$, the GE receives its embedding and the graph structure and computes its node representation:

$$\mathbf{h}_i^t = GE_t(\mathbf{n}_i, G_t),$$
$$\mathbf{h}_i^r = GE_r(\mathbf{n}_i, G_r)$$

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

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

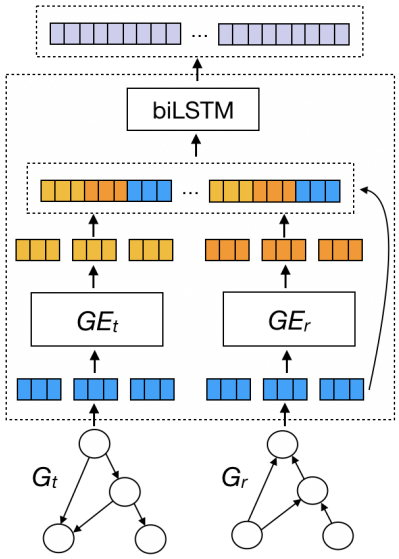

Figure 1: Encoder architecture. The encoder receives the two representations of the AMR graph and generates richer node representations that are used in the decoder.

The final representation $\mathbf{r}_i$ is employed in a sequence input of a bidirectional LSTM [25]. 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:

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

where $LSTM_f$ is a forward LSTM and $LSTM_b$ is a backward LSTM. Then we obtain the final hidden states by concatenating them as $\mathbf{h}_i = [\overrightarrow{\mathbf{h}}_i \,\|\, \overleftarrow{\mathbf{h}}_i]$, which saves the information of both the preceding and the following nodes. Figure 1 shows the proposed encoder architecture.

Finally, an attention-based decoder is leveraged to generate sentences, attending to edges and nodes hidden representations. Previous work [2, 5, 8] use anonymization to handle names and rare words, alleviating the data sparsity. Differently, to avoid repetition and to address out-of-vocabulary issues for rare target words, we employ copy and coverage mechanisms [26].

3.3 Graph Neural Networks

For each GE, we employ distinct strategies for neighborhood aggregation, adopting three GNNs: Gated Graph Neural Networks (GGNN, Li et al. [17]), Graph Attention Networks (GAT, [30]) and Graph Isomorphic Networks (GIN, [32]).

Gated Graph Neural Networks. GGNN employs gated recurrent units to encode node representations. In particular, the l-th layer

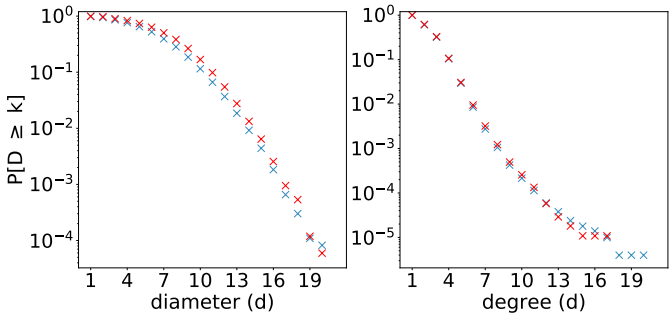

Figure 2: Distribution of the AMR graph diameter (left) and node degree (right) in the training set for LDC2015E85 (red) and LDC2017T10 (blue) datasets.

of a GGNN is calculated as:

$$\mathbf{h}_i^{(l)} = \text{GRU}\Big(\mathbf{h}_i^{(l-1)}, \sum_{j \in \mathcal{N}(i)} \mathbf{W}_1 \mathbf{h}_j^{(l-1)}\Big)$$

where $\mathcal{N}(i)$ is the immediate neighborhood of v_i and GRU is a gated recurrent unit [7].

Graph Attention Networks. GAT applies attentive mechanisms to improving the exploitation of non trivial graph structure. They encode node representations by attending over its 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:

$$\alpha_{i,j} = \frac{\exp\Big(\sigma\Big(\mathbf{a}^\top [\mathbf{W}_2 \mathbf{h}_i^{(l-1)} \,\|\, \mathbf{W}_2 \mathbf{h}_j^{(l-1)}]\Big)\Big)}{\sum_{k \in \mathcal{N}(i)} \exp\Big(\sigma\Big(\mathbf{a}^\top [\mathbf{W}_2 \mathbf{h}_i^{(l-1)} \,\|\, \mathbf{W}_2 \mathbf{h}_k^{(l-1)}]\Big)\Big)}$$

where σ is the activation function and $\|$ denotes concatenation. $\mathbf{W}_1$, $\mathbf{W}_2$ and $\mathbf{a}$ are model parameters.

Graph Isomorphic Networks. GIN is a GNN as powerful as the WL test [31] in representing isomorphic and non-isomorphic graphs with discrete attribute. 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).

Each of these GNNs applies different neighborhood aggregation and has achieved expressive results on many graph tasks [17, 30, 32].

4 EXPERIMENTS

4.1 Data

We use two AMR corpora, LDC2015E85 and LDC2017T10 with the default split of 16833/1368/1371 and 36521/1368/1371 instances for training, development and testing, respectively[2]. Figure 2 shows the distribution of AMR graph diameter and node degree for both datasets. The AMR graph structures are similar for most examples. Note that 90% of AMR graphs in both datasets have diameter less or equal than 11 and 90% of nodes have degree of 4 or less. Very 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 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 final node representation $\mathbf{r}_i$.

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

Model	BLEU	METEOR
LDC2015E86		
Song et al. (2018) [28]	23.28	30.10
Damonte et al. (2019) [8]	**24.40**	23.60
Cao et al. (2019) [5]	23.50	-
`seq2seq`	22.55 ± 0.17	29.90 ± 0.31
`graph2seq-GIN`	22.93 ± 0.20	29.72 ± 0.09
`graph2seq-GAT`	23.42 ± 0.16	29.87 ± 0.14
`graph2seq-GGNN`	**24.32** ± 0.16	**30.53** ± 0.30
LDC2017T10		
Song et al. (2018) [28]	24.86	31.56
Back et al. (2018) [2]	23.30	-
Damonte et al. (2019) [8]	24.54	24.07
Cao et al. (2019) [5]	26.80	-
`seq2seq`	22.73 ± 0.18	30.15 ± 0.14
`graph2seq-GIN`	26.90 ± 0.19	32.62 ± 0.04
`graph2seq-GAT`	26.72 ± 0.20	32.52 ± 0.02
`graph2seq-GGNN`	**27.87** ± 0.15	**33.21** ± 0.15

Table 1: BLEU and METEOR scores on the test set of LDC2015E86 and LDC2017T10 datasets.

4.2 Implementation details

We extract vocabularies from the training sets and the word embeddings are initialized from GloVe pretrained word embeddings [23] on Common Crawl. Hyper parameters are tuned on the development set of LDC2015E86 dataset. For GIN, GAT and GGNN graph encoders, we set the number of layers to 2, 5 and 5, respectively. To regularize the model, during training we apply dropout [29] between 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.

Our models are trained end-to-end to minimize the negative joint log likelihood of the target text vocabulary and the copied node, and are trained for 15 epochs with early stopping based on development accuracy. For our models and baseline we used a two-layer LSTM decoder. We use Adam optimization [13] as the optimizer with an initial learning rate of 0.001 and 20 as batch size. Beam search with beam size of 5 is used for decoding.

4.3 Results

We compare our models against several state-of-the-art results reported on these datasets [2, 5, 8, 28]. We call our models graph2seq-GIN (isomorphic encoder), graph2seq-GAT (graph-attention encoder) and graph2seq-GGNN (gated-graph encoder), according to the graph encoder utilized. As a baseline, we train an attention-based encoder-decoder model with copy and coverage mechanisms

AMR graph	(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)))
REF	China and Kyrgyzstan agreed in a joint communique that terrorism, separatism and extremism still pose major threats to regional security and stability.
seq2seq	In the joint communique, China and Kyrgyzstan still agreed to threaten terrorism, separatism, extremism and regional stability.
Song et. al (2018) [28]	In a joint communique, China and Kyrgyzstan have agreed to still be a major threat to regional security, and regional stability.
graph2seq-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 2: Examples of text generation from AMR graphs. REF is the reference sentence.

[26], and use a linearized version of the graph generated by depth-first traversal order as input. We use both BLEU [22] and METEOR [9] as evaluation metrics[3]. In order to mitigate effects of random seeds, we report the averages for 4 training runs of each model along with their standard deviation. Table 1 shows the comparison between our model, the baseline and other neural models on the test set of the two datasets.

For both datasets, our approach significantly outperforms the seq2seq baselines. In LDC2015E86, graph2seq-GGNN achieves a BLEU score of 24.32, 4.46% higher than Song et al. [28], 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 result to Damonte and Cohen [8], however we do not rely on preprocessing anonymisation avoid losing more precise semantic signals. In LDC2017T10, graph2seq-GGNN achieves a impressive BLEU score of 27.87, which is 13.56% higher than the best state-of-the-art model that does not employ external information [8].

We also outperform Cao and Clark [5] improving BLEU scores by 3.48% and 4%, in LDC2015E86 and LDC2017T10, respectively. In contrast to their work, we do no rely on (i) leveraging supplementary syntactic information and (ii) we do not require an anonymization pre-processing step. graph2seq-GIN and graph2seq-GAT have comparable performance on both datasets, with graph2seq-GIN having a slightly better performance in both BLEU and METEOR scores on LDC2017T10. Interestingly, graph2seq-GGNN has the better performance among our models. This suggests that graph encoders based on RNNs are very effective in text generation models. Our results empirically show that it is possible to incorporate more refined structured biases into the model, without relying on entity anonymization, positional embeddings and other artificial modifications in the topology of the graph.

Table 2 shows example outputs of seq2seq, Song et al. [28] and graph2seq-GAT. REF denotes the reference output sentence. While this is a single example, it suggests that computing dual node representation is beneficial for this task, which is supported by quantitative results.

[3]For BLEU, we use the multi-BLEU script from the MOSES decoder suite [14]. 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	57.6
GE_t + biLSTM	26.33	32.62	59.6
GE_r + biLSTM	26.12	32.49	59.6
GE_t + GE_r + biLSTM	27.37	33.30	61.7

Table 3: Ablation study on LDC2017T10 datasets. BLEU and METEOR scores on the development set of LDC2017T10 and number of model parameters (millions) including embeddings.

4.4 Ablation study

In Table 3, we report 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 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 (only with GE_t), as the neural model receives signals regarding the graph structure of the input. The dual representation (with GE_t and GE_r) 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).

5 CONCLUSION

We have studied the problem of generating text from AMR graphs. We have shown that our models exceed in producing text from AMR graphs, in comparison to state-of-the-art techniques [2, 5, 8, 28]. It overcomes their limitations by using a dual graph-encoder which jointly creates two parallel and adjuvant representations of the graph. We showed that encoding the dual representation enhances the overall performance. Finally, different models to generate graph representations tend to capture different properties [4], and choosing the most suitable graph encoder and architecture is important when considering different graph-based NLP tasks.

ACKNOWLEDGMENTS

This work has been supported by the German Research Foundation through the research training group "Adaptive Preparation of Information from Heterogeneous Sources" (AIPHES, GRK 1994/1).

REFERENCES

[1] 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*. Association for Computational Linguistics, Sofia, Bulgaria, 178–186. https://www.aclweb.org/anthology/W13-2322

[2] 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)*. Association for Computational Linguistics, Melbourne, Australia, 273–283. https://www.aclweb.org/anthology/P18-1026

[3] 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*. Association for Computational Linguistics, Nancy, France, 217–226. https://www.aclweb.org/anthology/W11-2832

[4] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. 2017. Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine* 34, 4 (July 2017), 18–42. https://doi.org/10.1109/MSP.2017.2693418

[5] 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 (NAACL-HLT)*.

[6] John Carroll and Stephan Oepen. 2005. High Efficiency Realization for a Wide-coverage Unification Grammar. In *Proceedings of the Second International Joint Conference on Natural Language Processing (IJCNLP'05)*. Springer-Verlag, Berlin, Heidelberg, 165–176. https://doi.org/10.1007/11562214_15

[7] 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)*. Association for Computational Linguistics, Doha, Qatar, 1724–1734. https://doi.org/10.3115/v1/D14-1179

[8] 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 (NAACL-HLT)*.

[9] 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*.

[10] 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*. Association for Computational Linguistics, San Diego, California, 731–739. https://doi.org/10.18653/v1/N16-1087

[11] 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*. Association for Computational Linguistics, Santiago de Compostela, Spain, 124–133. https://doi.org/10.18653/v1/W17-3518

[12] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 1024–1034. http://papers.nips.cc/paper/6703-inductive-representation-learning-on-large-graphs.pdf

[13] Diederick P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

[14] 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)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 177–180. http://dl.acm.org/citation.cfm?id=1557769.1557821

[15] 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 (NAACL-HLT)*.

[16] 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)*. Association for Computational Linguistics, 146–157. https://doi.org/10.18653/v1/P17-1014

[17] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. 2016. Gated Graph Sequence Neural Networks. *CoRR* abs/1511.05493 (2016).

[18] 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*. Association for Computational Linguistics, Santa Fe, New Mexico, USA, 1178–1190. https://www.aclweb.org/anthology/C18-1101

[19] 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*. Association for Computational Linguistics, Tilburg University, The Netherlands, 1–9. https://www.aclweb.org/anthology/W18-6501

[20] 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 (NAACL-HLT)*.

[21] Shashi Narayan and Claire Gardent. 2012. Structure-Driven Lexicalist Generation. In *Proceedings of COLING 2012*. The COLING 2012 Organizing Committee, Mumbai, India, 2027–2042. https://www.aclweb.org/anthology/C12-1124

[22] 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)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 311–318. https://doi.org/10.3115/1073083.1073135

[23] 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)*. Association for Computational Linguistics, Doha, Qatar, 1532–1543. https://doi.org/10.3115/v1/D14-1162

[24] Nima Pourdamghani, Kevin Knight, and Ulf Hermjakob. 2016. Generating English from Abstract Meaning Representations. In *Proceedings of the 9th International Natural Language Generation conference*. Association for Computational Linguistics, Edinburgh, UK, 21–25. https://doi.org/10.18653/v1/W16-6603

[25] Mike Schuster, Kuldip K. Paliwal, and A. General. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* (1997).

[26] 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)*. Association for Computational Linguistics, Vancouver, Canada, 1073–1083. https://doi.org/10.18653/v1/P17-1099

[27] 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 (March 2019), 19–31. https://doi.org/10.1162/tacl_a_00252

[28] 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)*. Association for Computational Linguistics, Melbourne, Australia, 1616–1626. https://www.aclweb.org/anthology/P18-1150

[29] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* 15, 1 (Jan. 2014), 1929–1958. http://dl.acm.org/citation.cfm?id=2627435.2670313

[30] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *International Conference on Learning Representations* (2018). accepted as poster.

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

[32] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *International Conference on Learning Representations*.

[33] Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, Michael Witbrock, and Vadim Sheinin. 2018. Graph2Seq: Graph to Sequence Learning with Attention-based Neural Networks. *arXiv preprint arXiv:1804.00823* (2018).