Improving Language Generation from Feature-Rich Tree-Structured Data with Relational Graph Convolutional Encoders

***†Xudong Hong, †Ernie Chang, †Vera Demberg**
*Dept. of Computer Vision and Machine Learning, MPI Informatics
†Dept. of Language Science and Technology, Saarland University
{xhong,cychang,vera}@coli.uni-saarland.de

Abstract

The Multilingual Surface Realization Shared Task 2019 focuses on generating sentences from lemmatized sets of universal dependency parses with rich features. This paper describes the system design and the results of our participation in the deep track. The core innovation in our approach is to use a graph convolutional network to encode the dependency trees given as input. Upon adding morphological features, our system achieves the second rank in the deep track without using data augmentation techniques or additional components (such as a re-ranker).

1 Introduction

The goal in the Multilingual Surface Realization Shared Task 2019 (MSR'19) is to generate fluent text from Universal Dependencies (UD) structures. The task makes available UD-annotated resources in 11 languages for the shallow task, and three languages (English, Spanish, French) for the deep track. Developing surface generation systems that are largely language-independent is a central objective of the shared task (Mille et al., 2018). To generate sentences based on the UD structure and morphological features, recent neural approaches mainly adopt neural sequence-to-sequence architectures (Cabezudo and Pardo, 2018; Madsack et al., 2018; Elder and Hokamp, 2018). While representing the feature-rich data in a linearized manner proved to be a viable option, we argue that these linear sequences do not optimally exploit the input information. We therefore propose to encode the dependency trees using a graph convolutional network (GCN) and find that this GCN encoder leads to a substantial boost in performance, compared to a sequential encoder.

The datasets in the deep track consist of semantic representations induced from syntactic dependency parses, see Figure 1 for an example. This task is reflects the information that's realistically available in real-world natural language generation task.

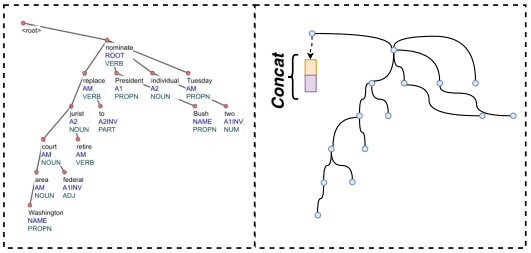

Figure 1: An example of a UD structure with concatenated feature embeddings from the MSR'19 deep task.

Our method works as follows: We first apply delexicalization to the datasets, replacing rare tokens with placeholders. Next, encode the dependency trees using graph representation learning techniques (Li et al., 2015; Xu et al., 2018a), in order to improve the encoding of structured data within the encoder-decoder architecture. Our model hence learns a mapping between graph inputs and sequence outputs. Our ablation study in the evaluation demonstrates that encoding UD structure in this manner does embed additional semantic information and subsequently improves the performance across the three languages available for the deep track (i.e. English, French, and Spanish). Finally, we use an LSTM decoder with copy mechanism and attention to generate surface text.

Our contributions are as follows:

1. We show that a GCN encoder for UD input structures outperforms sequential encoders.
2. We propose to use a variant of relational GCN (R-GCN) to better represent edge labels in the graph, and show that this boosts overall performance.
3. We show that structural encoding with the GCN benefits all three languages in the task.

Proceedings of the 2nd Workshop on Multilingual Surface Realisation (MSR 2019), pages 75–80
Hong Kong, China, November 3rd, 2019.
https://doi.org/10.18653/v1/P17

2 Related Work

2.1 Neural NLG

Systems proposed as part of the Surface Realization Shared Task 2018 are largely sequence-to-sequence models targeting the Shallow Task. Most systems in the past contain two separate components: 1) preprocessing of the UD dataset, and the 2) neural generator with the encoder-decoder architecture.

Most neural generators combine features by concatenating the aligned feature sequences and feed them as a single sequence into the neural generator (Elder and Hokamp, 2018; Madsack et al., 2018). In these systems, a pre-trained embedding is typically used to represent each lemmas, before concatenated with embeddings of surface-level morphological categories and dependency relations. A form of Recurrent Neural Network (RNN) are utilized to map the input to a latent space, and another RNN then decodes into target output. Examples of common RNN usage include Long Short-Term memory (LSTM) or the Bi-directional LSTM as used in Elder and Hokamp (2018); Madsack et al. (2018).

2.2 Graph-to-text Generation

Considering the fact that a dependency tree is a special case of a directed acyclic graph, surface realization is a graph-to-text generation tasks. Graph neural networks have been successfully applied to different graph to text generation task like SQL to text generation (Xu et al., 2018b), AMR-to-text generation (Beck et al., 2018) and semantic machine translation (Song et al., 2019). LSTM can be modified to model graph-level information (Song et al., 2018). Graph Convolutional Networks (GCN), originally designed for semi-supervised learning of node representations in graphs (Kipf and Welling, 2017), explicitly exploit tree structure data and outperform LSTM and TreeLSTM on AMR-to-text generation (Damonte and Cohen, 2019). To also model different types of edges in graphs, Relational Graph Convolutional Networks (R-GCN) represent each type of edge with a corresponding parameter matrix (Schlichtkrull et al., 2018). We leverage the R-GCN by grouping in-edge and out-edge together and apply to a graph-to-text generation task.

3 Our Approach

3.1 Feature Representations

The input format of the MSR'19 deep track is multi-source in the sense that each type of feature corresponds to a sequence of features, i.e., part-of-speech tags (POS), morphological features etc. As shown in Figure 1, we transform the tree-structured data into a graph. We construct node representations by simply concatenating token and its features. Then we use an embedding matrix to map the representations into low-dimensional vector space.

To handle rare words in input tokens, we firstly perform delexicalization for all datasets as follows:

1. Replace tokens that have part-of-speech tags of *NAME, PROPN, NUM* and *X* with placeholders jointly indexed by the number of head and the number of entities.

2. Build a dictionary from placeholders to original tokens for each input-output pair.

After obtaining the model output, we lexicalize the text by looking up each generated placeholder in the corresponding dictionary and insert the original token.

For our official submission to the shared task, we did not make use of features, in order to see whether the dependency tree is informative enough for surface realization. However, we performed additional experiments to show the effectiveness of GCN encoder with selected concatenated features, see Table 1.

3.2 Model

The graph-to-text generation task has a directed acyclic graph as input $G = \{V, E\}$, where V is a set of nodes and E is a set of directed edges e between nodes. In this paper, a node is an embedding vector containing a token and its features. An edge is the dependency relation between two nodes. The output Y is a sequence of tokens which form a sentence expressing the input. We extend the architecture by Marcheggiani and Perez-Beltrachini (2018) which combines a graph convolutional encoder and attentional LSTM decoder as described in Figure 2.

Graph Convolutional Encoder We use R-GCN, a variant of GCN (Schlichtkrull et al., 2018) assigning parameters for edges in a graph, to model

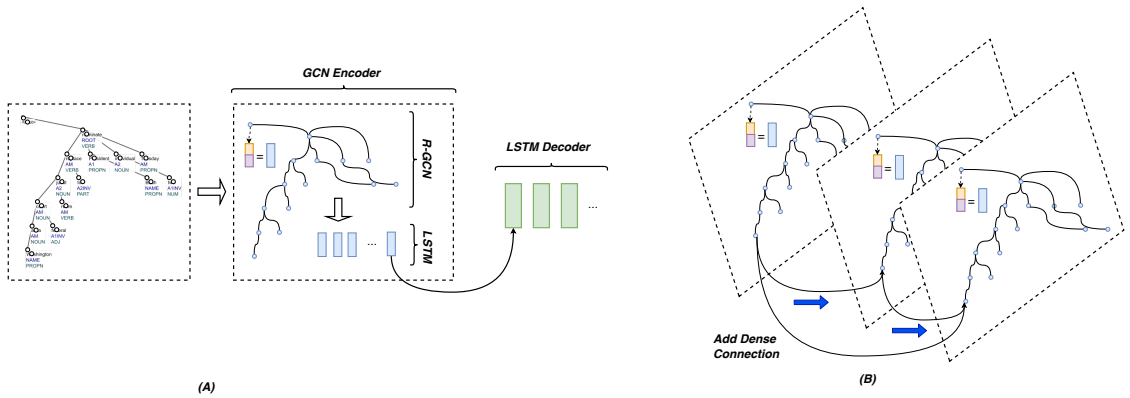

Figure 2: **(A)**: depicts the conceptual relationship between the GCN encoder and LSTM decoder. **(B)**: Addition of a dense layer is analogous to adding extra connections between multiple layers of graphical representation.

graph-structure input explicitly. Given a directed graph G, we represent each node with an embedding vector $\mathbf{x}_v \in \mathbb{R}^d$. Then the l-th R-GCN layer compute the hidden representation for node v in $(l+1)$-th layer as follows:

$$\mathbf{h}_v^{l+1} = f(\mathbf{W}\mathbf{h}_v^l + \sum_{u \in N(v)} \mathbf{W}_e \mathbf{h}_u^l) \quad (1)$$

where $\mathbf{W}, \mathbf{W}_e \in \mathbb{R}^{d \times h}$ and $e \in E$. f is the linear rectifier (ReLU), a non-linear activation function. $N(v)$ is the set of all neighbours of node v. This design is over-parameterized and there is no parameter sharing between similar edge labels. Therefore we redesign the update rule to:

$$\mathbf{h}_v^{l+1} = f(\mathbf{W}\mathbf{h}_v^l + \sum_{u \in N(v)} \mathbf{W}_{dir(e)} \mathbf{h}_u^l \circ \mathbf{r}_e) \quad (2)$$

where "$\circ$" is the Hadamard production, $\mathbf{W}_{r(u,v)} \in \mathbb{R}^{d \times h}$, $dir(e) \in \{in, out\}$ represents direction of the edge $e_{u,v}$ and $\mathbf{r}_e \in \mathbb{R}^h$ is an embedding vector of the label of $e_{u,v}$.

Each layer aggregates the direct neighbours of each node. To model neighbours of neighbours, we stack L GCN layers where L is set to the average radius of all graphs (here, average depth of all trees). Stacking GCN into deep neural networks could lead to gradient vanishing problem, thus we add residual connections (He et al., 2016) or dense connections (Huang et al., 2017) for each layer.

LSTM Decoder We apply stacked LSTM layers (Hochreiter and Schmidhuber, 1997) as the decoder on top of the GCN. The first layer is an input-feed LSTM (Luong et al., 2015) that aggregates the hidden representations of nodes into one hidden vector $\mathbf{h}_C$ for the whole graph. The second LSTM layer decodes the hidden vector and generates the representations of output token at each time step. We use global attention (Luong et al., 2015) to re-weight the hidden representations from the first layer and merge them into a global hidden vector $\mathbf{h}_G$. In order to generate the placeholder directly from the input, we apply the copying mechanism (Gu et al., 2016), which is effective when using lexcalization. The probability of token y_t conditioned on input G and previous token $y_{1:t-1}$ is obtained by applying a softmax layer on the decoder output as $P(y_t|y_{1:t-1}, G) = softmax(g(\mathbf{h}_G, \mathbf{h}_C))$, where g is a perceptron. The model is trained to maximize the likelihood function $L = \prod_{|Y|}^{t=1} P(y_t|y_{1:t-1}, G)$.

Encoder	BLEU	NIST	DIST
GCN (*)	23.0	6.88	42
LSTM	28.8	8.13	44.48
BiLSTM	31.2	8.53	46.86
GCN	35.9	8.73	52.86
R-GCN (*residual*)	39.81	9.24	55.45
R-GCN (*dense*)	**41.01**	**9.43**	**56.49**

(*) denotes system without morphological features, which is also our official submission to the shared task.

Table 1: Ablation study: results of models on the MSR'19 validation set of UD English EWT (enewt-ud-dev) corpus. We compare different encoders while keeping decoder constant, i.e., LSTM decoder with copy mechanism and coverage attention. For beam search we maintain a constant use of blocking 3-gram.

3.3 Experiments

We built our system on a variant of OpenNMT-py (Klein et al., 2017) from Marcheggiani and Perez-Beltrachini (2018) with customized encoders. We construct the training and validation datasets by concatenating corresponding splits of all available corpora for each language. We stack 4 R-GCN

Encoder	Output
BiLSTM	President Bush threw two members to replace manufacturers in the Washington area to replace manufacturers in federal nations.
GCN	In Tuesday, President Bush commissioned two connections to replace the federal individual of federal statements in the Washington area.
RGCN	In Tuesday, President Bush nominated two individuals to replace jurist trials to the Washington area.
Gold	President Bush on Tuesday nominated two individuals to replace retiring jurist on federal courts in the Washington area.

Table 2: Comparison of outputs from systems with encoder variants given the graph in Figure 1 as input. We highlighted obvious erroneous blocks of text for contrast. Note that the only variants are the encoders, all other configurations remain the same.

	Dev		Test	
Corpus	**GCN (*)**	**R-GCN (dense)**	**GCN (*)**	**R-GCN (dense)**
en_ewt-ud	23.0	41.01	23.35	18.37
en_gum-ud	17.71	34.47	17.97	14.6
en_lines-ud	18.32	12.7	20.96	14.89
en_partut-ud	18.54	35.3	17.19	12.85
es_ancora-ud	21.09	37.2	18.59	36.85
es_gsd-ud	20.56	33.39	18.69	35.92
fr_gsd-ud	20.48	35.12	15.83	10.65
fr_partut-ud	19.16	33.57	14.06	6.07
fr_sequoia-ud	21.07	34.49	18.52	10.22
en_pud-ud	-	-	18.11	12.31
en_ewt-Pred-HIT	-	-	22.42	39.05
en_pud-Pred-LATTICE	-	-	17.3	35.85
es_ancora-Pred-HIT	-	-	21.1	37.2

(*) denotes our submission to the shared task, which doesn't use morphological features

Table 3: Evaluation of our submissions to MSR'19 **Deep Task** across all corpora on both Test and Dev sets. Numbers are BLEU scores.

layers with dense connections as encoder and train the model with dropout rate of 0.5. We perform early stopping when the training accuracy is higher than the validation accuracy and choose the checkpoint before over-fitting for evaluation.

4 Results and Analysis

4.1 Encoder Model Selection

As indicated in Table 1, we compare R-GCN with different encoders. Systems are evaluated on the validation set of UD English EWT (enewt-ud-dev) corpus. With the same linearized inputs, we began with a LSTM encoder before moving on to bi-directional LSTM (BiLSTM). With 2.4 BLEU points improvement, BiLSTM appeared to be the option in terms of sequential encoder. Next, we employed the variant of GCN by Marcheggiani and Perez-Beltrachini (2018) with four fully-connected layers. We observed that this change gave an additional 4.7 BLEU points boost, which outperforms sequential encoders significantly. We then compare our R-GCN model to the GCN, which obtains additional 3.91 BLEU points. We further add dense conenctions to R-GCN, termed the *dense*, that eventually result in 41.01 BLEU points on the validation set. This was an overall 12 BLEU points improvement from the initial LSTM encoder.

4.2 Ablation Study: Decoder

We intend to investigate if the LSTM decoder can be further modified for improvement. Two of such changes are the **copy mechanism** and **coverage**

attention. The copy mechanism was shown to be beneficial in numerous similar tasks such as data-to-text generation (Li and Wan, 2018). With the addition of copy mechanism while keeping the encoder unchanged, an average of 1 BLEU score improvement can be observed; reusing the global attention for copying mechanism gave the system another 5 BLEU point boost.

4.3 Discussion

Our analysis shows that structural encoding of the UD trees leads to substantial improvements in performance. It also shows that including morphological features is crucial to performance of the surface realizer — without these features, we observe many errors in tense and agreement.

We also analyzed the system outputs to look for evidence to substantiate the intuition that a structural encoder can better represent the a priori linguistic information. One of such examples is shown in Table 2, where we observe fluency improvements going from BiLSTM encoder to GCN, and finally to R-GCN, where an overall improvement in fluency is conspicuous.

We report the results of our submissions in Table 3. Comparing to the validation results, **GCN(*)** trained without morphological features performs similarly across validation and test datasets of each corpus, however **R-GCN(dense)** has a significant drop from validation to test and experiences over-fitting. Importantly, we notice substantial BLEU rise and drop going from **GCN(*)** to **R-GCN(dense)** on the test datasets. We postulate that addition of relational modeling of edges (R-GCN) on top of rich features constrain the model to learn specific subset of a priori linguistic structures, thereby mitigating the overall performance.

5 Conclusion

We have shown that without additional modules such as re-ranker or data augmentation, the traditional encoder-decoder architecture can still be competitive by exploiting the existing structural input information. For future work, we intend to see if the performance can be further improved with pre-trained language models such as GPT-2 (Radford et al.).

Acknowledgements

This research was funded in part by the German Research Foundation (DFG) as part of SFB 1102 "Information Density and Linguistic Encoding" and SFB 248 "Foundations of Perspicuous Software Systems". Xudong Hong is supported by International Max Planck Research School for Computer Science (IMPRS-CS) of Max-Planck Institute for Informatics (MPI-INF). We sincerely thank the anonymous reviewers for their insightful comments that helped us to improve this paper.

References

Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-Sequence Learning using Gated Graph Neural Networks.

Marco Antonio Sobrevilla Cabezudo and Thiago Pardo. 2018. Nilc-swornemo at the surface realization shared task: Exploring syntax-based word ordering using neural models. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 58–64.

Marco Damonte and Shay B. Cohen. 2019. Structural Neural Encoders for AMR-to-text Generation. pages 3649–3658.

Henry Elder and Chris Hokamp. 2018. Generating high-quality surface realizations using data augmentation and factored sequence models. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 49–53.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Iclr*, pages 1–11.

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.

Liunian Li and Xiaojun Wan. 2018. Point precisely: Towards ensuring the precision of data in generated texts using delayed copy mechanism. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1044–1055.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2015. Gated Graph Sequence Neural Networks.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Andreas Madsack, Johanna Heininger, Nyamsuren Davaasambuu, Vitaliia Voronik, Michael Käufl, and Robert Weißgraeber. 2018. Ax semantics submission to the surface realization shared task 2018. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 54–57.

Diego Marcheggiani and Laura Perez-Beltrachini. 2018. Deep graph convolutional encoders for structured data to text generation. *arXiv preprint arXiv:1810.09995*.

Simon Mille, Anja Belz, Bernd Bohnet, Yvette Graham, Emily Pitler, and Leo Wanner. 2018. The first multilingual surface realisation shared task (sr18): Overview and evaluation results. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 1–12.

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

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling Relational Data with Graph Convolutional Networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10843 LNCS(1):593–607.

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. pages 1616–1626.

Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, Michael Witbrock, and Vadim Sheinin. 2018a. Graph2Seq: Graph to Sequence Learning with Attention-based Neural Networks. pages 1–18.

Kun Xu, Lingfei Wu, Zhiguo Wang, Mo Yu, Liwei Chen, and Vadim Sheinin. 2018b. Sql-to-text generation with graph-to-sequence model. *arXiv preprint arXiv:1809.05255*.