Exploiting Graph Neural Networks with Context Information for RDF-to-Text Generation

Hanning Gao*
Central China Normal University
gaohn@mail.ccnu.edu.com

Lingfei Wu*
IBM Research
lwu@email.wm.edu

Po Hu†
Central China Normal University
phu@mail.ccnu.edu.com

Fangli Xu
Squirrel AI Learning
lili@yixue.us

ABSTRACT

The task of RDF-to-Text generation is to generate a corresponding descriptive text given a set of RDF triplets. Most of the previous approaches either cast this task as a sequence-to-sequence problem or employ graph-based encoders for modeling RDF triplets and decode a text sequence. However, none of these methods can explicitly model both global and local structure information within and between the triplets. Furthermore, they did not exploit the target text as an important additional context for modeling complex RDF triplets. To address these issues, we propose to jointly learn local information and global structure information via combining graph encoder and graph-based triple encoder for the input triplets. Furthermore, we also exploit Seq2Seq-based auto-encoder to leverage target text as the context to supervise the combination of input encoders. Experimental results on the WebNLG dataset show that our proposed model outperforms the state-of-the-art baselines.

CCS CONCEPTS

• **Computing methodologies** → Neural networks.

KEYWORDS

Graph neural networks, RDF-to-Text generation, Auto-encoder

*Both authors contributed equally to this research.
†Corresponding author

KDD Workshop DLG '19, August 05, 2019, Anchorage, Alaska, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/1122445.1122456

ACM Reference Format:
Hanning Gao, Lingfei Wu, Po Hu, and Fangli Xu. 2019. Exploiting Graph Neural Networks with Context Information for RDF-to-Text Generation. In *Proceedings of DLG '19: The First International Workshop on Deep Learning on Graphs: Methods and Applications (KDD Workshop DLG '19).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/1122445.1122456

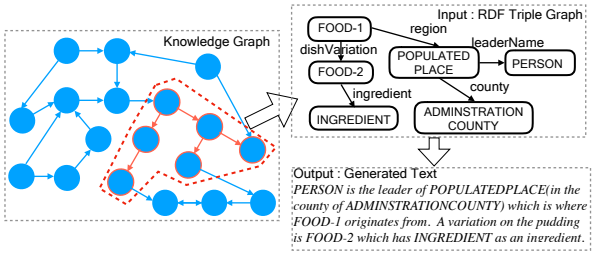

Figure 1: A knowledge graph formulated by a set of RDF triplets with generated text descriptions.

1 INTRODUCTION

RDF-to-text generation is a challenging task in natural language generation (NLG), which is to transform a set of RDF triplets into informative and faithful texts. Figure 1 illustrates an example of an input RDF triple graph with generated text. It has many fact-aware applications such as knowledge-based question answering, entity summarization, and data-driven news generation.

Traditionally, a RDF-to-text generation system mainly focused on the pipeline process of content selection [4] and surface realization [2] with human-crafted features. However, a critical issue for error propagation has been largely overlooked, which does harm to the quality of generated text. Recently, as end-to-end deep learning has made great progress, RDF-to-text generation has achieved promising performance by using various sequence-to-sequence (Seq2Seq) models [6, 11, 23] where RDF triplet elements are processed and concatenated into a sequence.

However, simply transforming the RDF triplets into a sequence may lose important higher-order information. Since

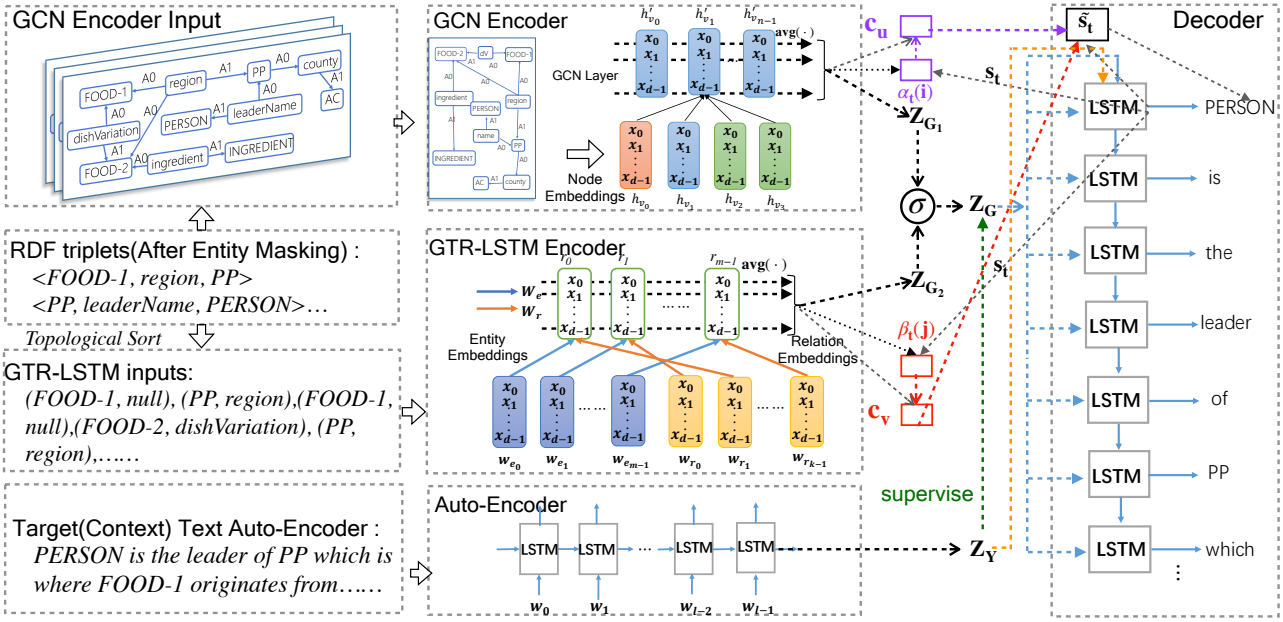

Figure 2: The framework of the combined encoder model with target text auto-encoder.

RDF triplets can be naturally represented as a knowledge graph, two graph-based approaches are recently proposed for RDF-to-text generation. Trisedya et al. [22] presented a graph-based triple encoder to capture both intra-triple and inter-triple entity relationships. However, since the encoder block is still based on recurrent neural networks, it often fails to capture rich local structure information between entities and relationships. On the other hand, Marcheggiani and Perez-Beltrachini [17] proposed a graph encoder based on a modified Graph Convolutional Networks (GCN) [13], which directly encodes graph-structured RDF triplets and decodes a text sequence. However, this graph-to-sequence (Graph2Seq) model usually focuses on local structure information of the graph and thus may fail to capture global information within and between the triplets to better capture rich local context information.

To address the aforementioned issues, we propose a novel neural network architecture by exploiting graph-based neural networks with context information for RDF-to-text generation. To this end, we first present to exploit both power of graph encoder and graph-based triple encoder to jointly learn structure information locally and globally and then leverage separate attentions to better decode the text sequence. The combined encoders can focus on multiple perspectives of the input RDF graph. The GCN encoder could explicitly model the local structure information between the intra-triplet relationships, while GTR-LSTM mainly focuses on modeling global long-range dependency information between the inter-triplet relationships. We further exploit a Seq2Seq-based auto-encoder with a single-layer bidirectional Long-Short Term Memory Networks (LSTM) [9] to leverage target text. The decoded target text could be considered as the auxiliary context to supervise the combination of graph and graph-based encoders for better representing complex RDF inputs. Similar ideas have also been exploited in [16, 21] for neural machine translation and text summarization tasks.

To summarize, we highlight our main contributions as follows:

- We propose a novel graph-based encoder model by combining GCN encoder and graph-based triple encoder (GTR-LSTM) for explicitly modeling the multiple perspectives of the input RDF triplets.
- We enhance our graph-based encoder model with a Seq2Seq-based auto-encoder to leverage target text as the auxiliary context to supervise our combined graph-based encoder.
- The experiment results on the WebNLG dataset corroborate the advantages of our model over state-of-the-art models in BLEU and METEOR metrics.

2 OUR PROPOSED MODEL

Formally, we define the RDF-to-text generation task as follows. The input side includes a set of triplets, which is denoted as $S = \{t_1, t_2, ...t_n\}$ where t_i is a triplet consisting of subject s_i, relationship r_i and object o_i. The aim is to generate a set of natural language sentences $Y = \langle w_1, w_2, ...w_T \rangle$ that represent the correct and concise semantics of entities and their relationships in the given input. To fit for graph

encoders, all the input triplets are represented as a directed labeled graph $G = (V, E)$ as a whole where V represents all of the nodes and E denotes a set of edges between the nodes in V. In this section, we introduce how to combine the two graph encoders we adopt and the target text auto-encoder.

Combined Encoders

We propose a combined strategy to integrate a GCN encoder and a GTR-LSTM triple encoder for the input RDF triplets, which is expected to jointly learn the local and global structure information of the RDF triplets input.

GTR-LSTM Triple Encoder. For the graph G, V is a set of entity nodes(v_i), and E is a set of directed edges in which $e_{i,j}$ represents the relationship between v_i and v_j. The representation of v_j is computed as: $r_j = f(r_i, v_j, e_{i,j})$, where v_i is a parent node of v_j in the graph and $f(\cdot)$ is a single GTR-LSTM unit, following the implementation of Trisedya et al. [22]. For the nodes having multiple $(v_k, e_{k,j}, v_j)$ triples, v_j has multiple representations $\{r_{j_i}, r_{j_k}, ...\}$ preserved in a final set of entity node representations $R_{GTR-LSTM}$.

Graph Convolutional Encoder. Each node $v \in V$ is represented with $h_v{}'$ at current layer and h_v at previous layer. The node embedding can be updated as follows

$$h_v{}' = ReLU(\sum_{u \in N(v)} g_{u,v}(W_{dir(u,v)} h_u + b_{lab(u,v)})) \quad (1)$$

where $N(v)$ is the neighbour nodes of v, and $W_{dir(u,v)}$ is a direction-specific parameter matrix, in which $dir(u, v) \in \{in, out, loop\}$ [18]. And $b_{lab(u,v)} \in R^d$ is an embedding of the label of the edge. $g_{u,v}$ are learned scalar gates which weight the importance of each edge. GCN layers are stacked to multiple layers using residual connections [8] and dense connections [10].

Both encoders generate a set of node representations. In general, $R_{GCN} = \{h_0, h_1, h_2...\}$ captures the local structure information within the RDF triplets better while $R_{GTR-LSTM} = \{r_0, r_1, r_2, ...\}$ mainly focuses on the global structure information between the RDF triplets.

This motivates us to obtain a graph embedding G_i by combining R_{GCN} and $R_{GTR-LSTM}$ through applying an average pooling on $R_i \in \{R_{GCN}, R_{GTR-LSTM}\}$.

$$Z_{G_i} = avg(R_i) \quad (2)$$

where $avg(\cdot)$ denotes the average operator. The combined graph embedding Z_G is computed as,

$$Z_G = \sigma(W_g[Z_{G_1}; Z_{G_2}] + b_g) \quad (3)$$

where σ is a non-linearity(ReLU), W_g and b_g are learnable parameters. Z_G is fed into the decoder at initial hidden state and input. We then further apply the separate attentions by computing the align weights vector at decoding time step t as follows:

$$\alpha_{t(i)} = \frac{\exp(score(h_i, s_t))}{\exp(\sum_{k=1}^{V} score(h_k, s_t))} \quad (4)$$

$$\beta_{t(j)} = \frac{\exp(score(r_j, s_t))}{\exp(\sum_{k=1}^{M} score(r_k, s_t))} \quad (5)$$

where $h_i \in R_{GCN}$ and $r_j \in R_{GTR-LSTM}$. And $V = |R_{GCN}|$ and $M = |R_{GTR-LSTM}|$ are the lengths of representations sequence. The $score(\cdot)$ function estimates the similarity of h_i, r_j and s_t. Then, we compute GCN-level context vector c_u and GTRLSTM-level context vertor c_v as follows:

$$c_u = \sum_{i=1}^{V} \alpha_{t(i)} h_i \quad (6)$$

$$c_v = \sum_{j=1}^{M} \beta_{t(j)} r_j \quad (7)$$

Next, we compute final attentional hidden state at this time step as:

$$\tilde{s_t} = \tanh(W_c \cdot [c_u; c_v; s_t] + b) \quad (8)$$

where W_c and b are learnable parameters. Finally, we employ the loss function with a negative log likelihood:

$$L_G = \frac{1}{T} \sum_{t=1}^{T} -\log P(y_t | y_{0:t-1}, Z_G) \quad (9)$$

Auto-encoder Supervision

The target texts are well-written and contain almost the same information as the input triplets. Therefore, we explicitly exploit the target text as the auxiliary context to supervise graph encoder learning. At the training phase, target text encoder compresses reference text Y into Z_Y. Except conditioned on Z_Y, the loss function L_{AE} is the same as equation 9. L_{AE} is computed as:

$$L_{AE} = \frac{1}{T} \sum_{t=1}^{T} -\log P(y_t | y_{0:t-1}, Z_Y) \quad (10)$$

The supervision of target text auto-encoder is realized by minimizing the distance loss between Z_G and Z_Y . L_{dis} is added into the loss function as regularization item:

$$L_{dis} = \frac{\lambda}{N_h} d(z_G, z_Y) \quad (11)$$

where N_h is the hidden size of target text encoder and $d(Z_G, Z_Y)$ is a L2 constraint. λ is a hyper-parameter.

Workflow of the Overall Framework

The overall architecture of our proposed model is depicted in Figure 2. The encoders receive differently preprocessed RDF triplets set as input. Both encoders generate a set of node representations R_i and a graph embedding Z_{G_i}. Then auto-encoder generates a representation Z_Y for the target text. Z_G and Z_Y are fed into the decoder as initial hidden state and input, respectively. Attention mechanism is applied on encoders outputs R_{GCN} and $R_{GTR-LSTM}$. The model is trained to optimize an overall loss function:

$$Loss = L_G + L_{AE} + L_{dis} \tag{12}$$

3 GRAPH CONSTRUCTIONS

In this section, we describe how to transform RDF triplets set to input graphs for two encoders.

Entity Masking. Entity masking can improve the generalization ability of our model. Considering that multiple entities in a set of triplets may belong to the same type, we assign an eid to each entity in the set. Therefore, each entity is replaced with its eid and type. For example, the entity "Bakewell pudding", know as "FOOD-1" in figure1, is replace by "ENTITY-1 FOOD" while the entity "Bakewell tart"("FOOD-2") is replace by "ENTITY-2 FOOD".

RDF Triplets to Graph. For GTR-LSTM triple encoder, we first choose a starting node with 0 in-degree and employ topological sort traversal algorithm to the graph. The traverse order of figure1 is (FOOD-1, POPULATEDPLACE(PP), FOOD-1, FOOD-2, PP, PERSON, PP, ADMINSTRATIONCOUNTY, FOOD-2, INGREDIENT). Each node receives one of its parent nodes embedding as a hidden state. For GCN encoder, we treat relations as additional nodes [18] and the new relation node is connected to the subject and object by two new relations(i.e., A0 and A1), respectively. The original triplet (FOOD-1, region, PP) is separated into two new triplets (region, A0, FOOD-1) and (region, A1, PP) with "region" being a node in the input graph.

4 EXPERIMENTS

Datasets

We use the WEBNLG dataset [5] for the task of mapping RDF-triples to text. An example of the dataset is a $\langle S, T \rangle$ pair, consisting of a source side triple set and a target side reference text. Here, one triple set may correspond to multiple reference texts. Each RDF triple is represented as <subject, relationship, object> , where the subject and object are constants or entities. The dataset [1] is avaliable. The whole dataset is divided into 18102 training pairs, 2495 validation pairs and 2269 testing pairs. As mentioned in section 3, to enhance the generalization capability of the model, we map the subjects and objects in the triple set to entity types using the officially provided dictionary and DBpedia lookup API[2].

Experimental Settings

We implement the baseline model and our proposed model based on OpenNMT [3] released by Klein et al. [14]. We use Adam[12] as optimization method with an initial learning rate 0.001 and learnable parameters are updated every 64 instances. We train 10000 steps(about 35 epochs) on GPU and evaluate on validation set every 500 steps. The word embeddings is sharing between source and target text. Copy mechanism[7] is used in every model.

Sequential Encoders We used a standard attention-based[1, 15] sequence-to-sequence model with one layer bidirectional LSTM encoder and one layer LSTM decoder as baseline. The RDF triplets are simply transformed into a sequence and fed into the encoder. The embeddings and hidden units are set to 300 dimensions.

GTR-LSTM Encoder The GTR-LSTM model consists of a GTR-LSTM encoder and one layer LSTM decoder. GTR-LSTM Encoder is re-implemented according to Trisedya et al. [22]. Both encoder and decoder embeddings and hidden units use 300 dimensions. Each node receives one of its parent nodes embedding as a hidden state and the concatenation of entity embedding and relation embedding as input. Since one node may have multiple parent nodes, this kind of nodes may have multiple representations preserved in a final set of entity node representations $R_{GTR-LSTM}$.

GCN Encoder The GCN models consist of a GCN encoder and one layer LSTM decoder. We follow the implementation in [17] and the code[4] is available. All encoder and decoder embeddings and hidden units use 300 dimensions. We extend the GCN-based model with copy mechanism [20]. We experimented on 2,3 and 4 layers with residual connection.

Combined Graph Encoders with Target Text Auto-encoder The combined model consists of a GCN encoder, a GTR-LSTM encoder, a target text auto-encoder and LSTM decoder. We then further apply the separate attentions on the combined graph encoders. All encoder and decoder embeddings and hidden units are 300 dimensions and copy mechanism is also a default setting.

Metrics and Results

For our experiments, we adopt the standard evaluation metrics of the WebNLG challenge, including BLEU [19] and METEOR [3]. BLEU looks at n-grams overlap between the

[1] https://gitlab.com/shimorina/webnlg-dataset/tree/master/webnlg_challenge_2017

[2] The query result of 'Bakewell pudding' using DBpedia lookup API is http://lookup.dbpedia.org/api/search/KeywordSearch?QueryString=Bakewellpudding.

[3] http://opennmt.net/

[4] https://github.com/diegma/graph-2-text

RDF Triples Input	(William Anders, dateOfRetirement, 1969-09-01) (William Anders, nationality, United States) (William Anders, occupation, Fighter pilot) (William Anders, birthPlace, British Hong Kong) (William Anders, was a crew member of, Apollo 8)
Reference Output	william anders is an american who was born in british hong kong . he became a fighter pilot and later a member of the crew on apollo 8 . he retired on 1 september 1969.
GCN	american william anders was born in british hong kong . he was a fighter pilot and a crew member of apollo 8 .
GTR-LSTM	william anders , a united states national , was born in united states . he was a fighter pilot and crew member on apollo 8 . he retired in 1969-09-01 .
Our Model	william anders is a united states national who was born in british hong kong . he was a fighter pilot and a crew member of apollo 8 . he retired in 1969-09-01 .

Table 1: Sample outputs of different models.

Model	BLEU	METEOR
Bi-LSTM	52.60	38.3
GTR-LSTM	54.54	39.9
GCN(2 Layers)	54.76	40.3
GCN(3 Layers)	54.93	40.3
GCN(4 Layers)	54.55	40.2
GCN(2 Layers) + GTR-LSTM+AE	**56.69**	**40.7**

Table 2: BLEU and METEOR on WebNLG test dataset.

Model	BLEU	METEOR
Full Model	**56.69**	**40.7**
-AE	55.31	40.2
-L_{dis}	56.22	40.4
-GCN	54.92	40.1
-GTR-LSTM	55.49	40.6

Table 3: Ablation study on WebNLG test dataset.

output and reference text with a penalty for shorter outputs. METEOR modifies the precision and recall computations, replacing them with a weighted F-score based on mapping unigrams and a penalty function for incorrect word order. For RDF-to-text generation, our proposed model is able to better encode the global and local graph structure of the input RDF triplets. For instance, our proposed model is about 2.0 BLEU points higher than those of the other baselines on the WebNLG dataset as shown in Table 2.

Qualitative Analysis

In addition, we further manually inspected the outputs of different models. As shown in Table 1, we found that the models involving GCN encoder perform better on covering correct relations between entities; target text auto-encoder and GTR-LSTM encoder perform better on generating texts associated with input context information among the RDF triplets. This result is expected since GCN encoder pays more attention to the local structure information and can predict the intra-triplet relationship more accurately and effectively. Meanwhile, GTR-LSTM mainly focuses on the inter-triplet relationships, which helps model long-range dependency and associate with more input context information.

Ablation Study

As shown in Table 3, there are four key factors in our proposed model that may affect the quality of generated text. The first two are the target text auto-encoder and L_{dis}. The first one is the target text auto-encoder, which will help to integrate target side context information. The second factor L_{dis} minimizes the distance between graph representation and text representation. The other two are the GCN encoder and the GTR-LSTM encoder, which encode the local and global information of the input triplets set. We separate each of these four components from the full model. Experiment results are listed in Table 3. The performance of both Full Model - GCN and Full Model - GTR-LSTM decreases by at least 1.77 and 1.20 BLEU points. This result is expected, since it is difficult for one single graph-based encoder to fully encode both global and local structure information completely. We also observe that integrating context information helps the graph encoder learn better semantic representations for the RDF triplets. Finally, the contribution of L_{dis} is shown to be marginal compared to the full model.

5 CONCLUSION

We propose a novel approach via exploiting graph-based neural networks with context information for RDF-to-text generation. Our approach jointly learns structure information locally and globally via the combination of a graph encoder and a graph-based triple encoder to learn intra-triplet and inter-triplet relationships. We further exploit Seq2Seq-based auto-encoder for the target text as the auxiliary context to supervise our combined encoder. The experiments show that our proposed model outperforms the state-of-the-art baselines. In future work, we plan to investigate the contribution of two graph encoders in a quantitative way and make further improvement on this model.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[2] Kees Van Deemter, Mariët Theune, and Emiel Krahmer. 2005. Real versus template-based natural language generation: A false opposition? *Computational Linguistics* 31, 1 (2005), 15–24.

[3] Michael Denkowski and Alon Lavie. 2011. Meteor 1.3: Automatic metric for reliable optimization and evaluation of machine translation systems. In *Proceedings of the sixth workshop on statistical machine translation*. Association for Computational Linguistics, 85–91.

[4] Pablo A Duboue and Kathleen R McKeown. 2003. Statistical acquisition of content selection rules for natural language generation. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*. Association for Computational Linguistics, 121–128.

[5] Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating Training Corpora for Micro-Planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada.

[6] 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*. 124–133.

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

[8] 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*. 770–778.

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

[10] 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*. 4700–4708.

[11] Glorianna Jagfeld, Sabrina Jenne, and Ngoc Thang Vu. 2018. Sequence-to-Sequence Models for Data-to-Text Natural Language Generation: Word-vs. Character-based Processing and Output Diversity. *arXiv preprint arXiv:1810.04864* (2018).

[12] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[13] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[14] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. In *Proc. ACL*. https://doi.org/10.18653/v1/P17-4012

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

[16] Shuming Ma, Xu Sun, Junyang Lin, and Houfeng Wang. 2018. Autoencoder as assistant supervisor: Improving text representation for chinese social media text summarization. *arXiv preprint arXiv:1805.04869* (2018).

[17] 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*. 1–9.

[18] Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. *arXiv preprint arXiv:1703.04826* (2017).

[19] 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*. Association for Computational Linguistics, 311–318.

[20] Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368* (2017).

[21] Linfeng Song, Daniel Gildea, Yue Zhang, Zhiguo Wang, and Jinsong Su. 2019. Semantic Neural Machine Translation using AMR. *arXiv preprint arXiv:1902.07282* (2019).

[22] 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)*, Vol. 1. 1627–1637.

[23] Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2017. Challenges in data-to-document generation. *arXiv preprint arXiv:1707.08052* (2017).