# RECYCLEGT - UNSUPERVISED GRAPH TO TEXT AND TEXT TO GRAPH TRANSLATION

**Jeffrey Cheng, Rami Sbahi, Ruoyu (David) Wu & Yuncong (Joe) Zuo**
Duke University
{jeffrey.cheng154,rami.sbahi,ruoyu.wu,yuncong.zuo}@duke.edu

## ABSTRACT

In our final project, we attempted to replicate and improve a graph-structure model. The translation between text and knowledge graph is a common problem in natural language processing and we propose ReCycleGT, a model that implements cycle training for both Text-to-Graph (T2G) and Graph-to-Text (G2T) tasks, and it can bootstrap from fully non-parallel graph and text data, and iteratively back translate between the two forms in order to train itself in an unsupervised manner. ReCycleGT also looks at supervised training in order to compare with other supervised models. Our code can be found at this repository.

## 1 INTRODUCTION

In knowledge representation and reasoning, a knowledge graph is a kind of graph-structure data model to integrate data. It offers a more intuitive representation of text semantics and it has broad usages in terms of visualizing and facilitating reading, along with many downstream tasks, like information retrieval and reasoning. There are mainly two questions of our interest: Graph-to-Text transformation (G2T) and Text-to-Graph transformation (T2G). We will be reproducing and improving upon the results of this paper.

G2T produces descriptive text that verbalizes the graphical data, while T2G extracts structures in the form of knowledge graphs from the text, so that all entities become nodes, and the relationships among entities form edges. These questions both see uses in a variety of NLP tasks. For instance, G2T can be used in querying tasks where the model is required to translate structured information to the user. T2G can be similarly used to extract information from a passage of text, for tasks such as reasoning and fact checking.

## 2 RELATED WORK

**Graph-to-text** For the graph-to-text generation task, the model takes in a knowledge graph and generates a corresponding natural language description. Recent efforts for this task mainly adopt transformer-based pretrained language models such as GPT-2 (Radford et al., 2019), BART (Lewis et al., 2020), and T5 (Raffel et al., 2020), which have achieved state-of-the-art results on the WebNLG 2017 dataset.

**Text-to-graph** The text-to-graph extraction task is typically split into two sub-tasks, which are Named Entity Recognition (NER) and Relation Extraction (RE). There are off-the-shelf tools that perform NER with good performance, and many models based on CNN (Nguyen & Grishman, 2015), RNN (Zhou et al., 2016), and BERT (Wang et al., 2019) have achieved high performance on RE tasks. There have also been attempts that perform NER and RE jointly (Lin et al., 2020; Kolluru et al., 2020), which have shown impressive performance since they can mitigate error propagation and leverage inter-dependencies between the tasks.

**Cycle training** The concept of leveraging the transitivity of two functions inverse to each other has been observed on a variety of tasks such as image style transfer (Zhu et al., 2017) and language style transfer (Jin et al., 2019). The paper we aim to reproduce (Guo et al., 2020) uses T5 as the G2T model and a BiLSTM for the T2G architecture and adopts an iterative back translation training scheme to improve both models.

# 3 Approach

## 3.1 Background

The model G2T: $\mathcal{G} \rightarrow \mathcal{T}$ takes as input a graph $g$ and generates a text sequence $\hat{t}$ that is a sufficient description of the graph. The model T2G: $\mathcal{T} \rightarrow \mathcal{G}$ takes as input a text sequence $t$ and extracts its corresponding graph $\hat{g}$, whose nodes are the entities and edges are the relations between two entities. For cycle training with iterative back translation, there are a variable $\boldsymbol{x}$ and a bijective mapping function $f(\cdot)$ that satisfy $\boldsymbol{x} = f^{-1}(f(\boldsymbol{x}))$. In our case, the functions G2T and T2G are inverses of each other. We want to align each text with its back-translated version and also each graph with its back-translated version with bijection, i.e., our objective is to collectively train the two models so that we have

$$t = \text{G2T}(\text{T2G}(t)) \qquad g = \text{T2G}(\text{G2T}(g))$$

We will denote the former as a "T-cycle" (as the prediction starts from Text) and the latter as a "G-cycle" (as the prediction starts from Graph).

## 3.2 Method

The supervised training of the T2G and G2T models is very standard. Given a dataset $D$ consisting of parallel data $(t, g)$, training the model amounts to finding parameters $\varphi^*, \psi^*$ that satisfy:

$$\varphi^* = \arg\max_{\varphi} \prod_{(t,g)\in D} p(g \mid t; \varphi) \qquad \psi^* = \arg\max_{\psi} \prod_{(t,g)\in D} p(t \mid g; \varphi)$$

For unsupervised training, we have a dataset $D_{\mathcal{T}}$ consisting of text and a dataset $D_{\mathcal{G}}$ consisting of graphs. We train the two models in parallel by first obtaining the predictions of the T2G model on a text batch $t \in D_{\mathcal{T}}$, creating a batch of synthetic graphs, denoted $T2G(t)$. Then, we feed the batch of synthetic graphs into the G2T model to obtain log probabilities of the predicted sentence, with which we compute negative log-likelihood (NLL) loss with the original gold text $t$, and update the G2T model parameters. We then feed in a graph batch $g \in D_{\mathcal{G}}$ and perform the same process, swapping the order of the two models, in order to update the T2G model parameters. Essentially, we approximate the parallel data by creating pairs $(t, T2G(t))$ and $(g, G2T(g))$ and train the two models by alternating inputs of batches of text and graphs.

## 3.3 Extensions

We perform some hyperparameter tuning in training our models, experimenting with 2 different learning rates and 2 different batch sizes to find our optimal models.

Furthermore, the original repository's code contained no indication that the cycle model used T5 as indicated in their paper. Thus, the implementation of T5 over the BiLSTM present in the original paper can be viewed as an extension of the original task.

# 4 Experiments

## 4.1 Datasets and Evaluation

The main dataset used was the WEBNLG 2017 dataset (Gardent et al., 2017) which is frequently used in text-to-graph and graph-to-text generation tasks. We chose to first use this dataset since its data is provided in parallel, allowing us to test our separate T2G and G2T models with supervised training. After the supervised testing, we created dataloaders to shuffle the data so that it was no longer parallel, and used the unparallel data in our cycle model.

WebNLG 2017 consists of approximately 13000 examples. We used a pre-trained entity extraction model to extract the entities in the text, and the resulting examples are json object consisting of text, relations, and entities. An example of a json object is given below.

```
{'relations': [['Aarhus', 'Airport'],
  'cityServed',
  ['Aarhus', ',', 'Denmark']]],
 'text': 'the <ENT_1> of <ENT_0> .',
 'entities': [['Aarhus', ',', 'Denmark',], ['Aarhus', 'Airport']]}
```

We split the WebNLG 2017 dataset into train.json, dev.json, and test.json datasets. test.json was used for evaluating all of our models. We evaluate T2G and G2T separately across all the data in this dataset with both supervised and unsupervised models. Throughout the training process, we do not use this data. Rather, we use train.json (for training) and dev.json (for evaluation each epoch).

For the T2G task, the metric used to judge performance is the micro and macro F1 scores based on the number of correct relation predictions between given entities. On the other hand, the G2T task performance is measured using BLEU, ROUGE, METEOR, and CIDEr evaluation metrics.

### 4.2 Model and Training Details

#### 4.2.1 T2G Model

The T2G model takes in a batch of tokenized sentences (which were computed using a precomputed vocabulary). Word embeddings are computed for the tokenized sentence. These word embeddings are fed through a bidirectional LSTM.

The forward function of the LSTM model outputs an $n \times n \times R$ matrix $m$, where $n$ is the maximum number of entities in the batch and $R$ is the number of possible relations between any two entities. In this step, we consider the words at the indices corresponding to the substituted entities and take an average over the LSTM outputs at those indices for each entity to obtain a "contextualized entity" for each entity. These "contextualized entities" (corresponding to the vertices of the graph) are then taken pairwise and passed through a multi-label classification layer to predict the relationship between them (corresponding to the edge of the graph).

This matrix $m$ gives the log-probabilities of any relation between two entities; i.e. $m[e_1][e_2][r]$ is the log probability of there being an $r$ relation between entities $e_1$ and $e_2$. This batch of matrices $m$ is the output of the forward function of T2G.

Then, by performing an argmax over the last dimension of $m$, the predict function of the T2G model outputs a batch of $n \times n$ matrices $x$.

$$x[e_1][e_2] = \arg\max_r m[e_1][e_2][r]$$

Each of these matrices $x$ describes the predicted relation between any two entities; i.e. $x[e_1][e_2]$ is the index of the predicted relation between entities $e_1$ and $e_2$ where $e_1 \leq e_2$. $x[e_1][e_2] = Q$ for all $e_1 > e_2$ in this matrix, where $Q$ is the index of no relation in the relation vocabulary. This batch of matrices $x$ is the output of the predict function of T2G.

The aim of the T2G model is to find the optimal parameter $\varphi^*$ that correctly encodes the text and predicts the graph. In unsupervised learning, this is measured loss between the predicted graph and the gold graph in a G-cycle.

$$\varphi^* = \arg\max_\varphi \prod_{(t,g)\in D} p(g \mid t; \varphi) = \arg\max_\varphi \prod_{t,g\in D} \prod_{i=0}^{n} \prod_{j=0}^{n} p(m[e_i][e_j][r] \mid v_i, v_j, t; \varphi)$$

To ensure that the inputs to each batch are a tensor, we preprocess each batch by padding sentences to be the same length with $\langle EMPTY \rangle$ tags and add dummy entities to each example that are eventually masked out during training.

In the T2G model, our BiLSTM model hidden size is 512 and consists of two layers (one for each direction). Each of the linear layers has input and output dimensions of 512 as well, excluding the last layer's output dimension which equals the number of relations in our tokenizer. We use the default Adam optimizer with a fixed learning rate of 0.002.

#### 4.2.2 G2T MODEL

The G2T model takes in a batch of linearized graphs and returns a batch of sentences (in dictionary / JSON object form, with 'text' and 'entities' keys).

Note that a linearized graph is a string of text with $\langle H \rangle$, $\langle R \rangle$, and $\langle T \rangle$ components. For instance, consider the following graph, $g$:

```
{'relations': [[['Aarhus', 'Airport'],
    'cityServed',
    ['Aarhus', ',', 'Denmark']]],
 'entities': [['Aarhus', ',', 'Denmark',], ['Aarhus', 'Airport']]}
```

The linearized version of this graph, Seq($g$), is as follows:

```
g2t: <H> Aarhus Airport <R> city Served <T> Aarhus , Denmark
```

In the case of multiple relations, these would be repeatedly appended to the end of the linearization (starting with $\langle H \rangle$).

This batch of linearized graphs is fed into the T5 model, which tokenizes this linearization and outputs a predicted sentence for each graph in the batch.

The aim of the G2T model is to find the optimal parameter $\psi^*$ that correctly encodes the graph and predicts the text. In unsupervised learning, this is measured as the loss between the predicted text and the gold text in a T-cycle.

$$\psi^* = \arg\max_{\psi} \prod_{(t,g)\in D} p(t \mid g; \varphi) = \arg\max_{\psi} \prod_{(t,g)\in D} p(t \mid \text{Seq}(g); \psi)$$

#### 4.2.3 ITERATIVE BACK TRANSLATION (CYCLE TRAINING)

T-CYCLE The T-cycle is given a batch of "gold" sentences (in dictionary / JSON object form, with 'text' and 'entities' keys). These sentences first must be preprocessed by substituting the $\langle ENT_k \rangle$ strings for the actual words and then converting the concatenated sentence into indices using a precomputed vocabulary. Note that in this step a different index is assigned to a word if a word is observed as part of an entity than if that word is observed outside of an entity. This batch of tokenized sentences is fed into the predict function of T2G to compute the batch of matrices $x$ which describe the predicted relations between any two entities, as described in 4.2.1. In post-processing, these matrices are converted to our ideal 'graph' form, which is a list of dictionaries (each formatted like the example JSON object, but only with 'relations' and 'entities' keys).

These dictionaries are then linearized in a pre-processing step and input into G2T. These linearized graphs are then fed into the G2T model, which outputs a batch of predicted sentences. These sentences are used to compute NLL loss by comparing to the batch of "gold" sentences. The parameters of G2T are updated based on the loss.backward() call.

G-CYCLE The G-cycle is given a batch of knowledge graphs $G$ (in dictionary / JSON object form, with 'relations' and 'entities' keys). These graphs are linearized in a pre-processing step and input into the G2T model, which outputs a batch of predicted sentences.

These sentences are then tokenized as in the pre-processing of T2G in the T-cycle step. These tokenized sentences are then inputted into the forward function of the T2G model, where we output the batch of predicted log probability matrices $m$.

Now, we convert the graphs in $G$ into $n \times n$ matrices, where $n$ is the maximum number of entities of any graph in the batch. These matrices are analogous to the matrix $x$ described in 4.2.1; they describe the true relations between any two entities in the graph and the bottom half (i.e. $e_1 \geq e_2$) of this matrix is set to $\langle EMPTY \rangle$. These serve as our "gold" graphs.

Then, we can compute the NLL loss is computed over the graphs in the batch by comparing the predicted log probabilities matrices to the "gold" graphs, ignoring any $\langle EMPTY \rangle$ indices. Finally, the parameters of T2G are updated based on the loss.backward() call.

GENERAL PROCESS In our iterative back-translation training process, we alternate between performing a T-cycle on a batch of text and a G-cycle on a batch of graphs. This was done for approximately $b = 408$ nonparallel batches of size 16, and then repeated for $c = 10$ epochs. We experimented with batches of size 8 and 32, as well, but found size 16 to be optimal.

### 4.3 CHALLENGES FACED IN REPLICATION

Creating the individual models for supervised T2G and G2T training were not that difficult. While it took some time to understand the architecture of the T2G model, its essence is simply a multiclass classification problem. The G2T model could be black-boxed using SimpleT5, which made training and inference extremely simple.

The main challenges came when trying to implement the unsupervised cycle training. From a replication standpoint, the original paper provided little to no detail of how the two models functioned in tandem. Specifically, in a T-cycle, after forming a batch of synthetic graphs from calling T2G's prediction function, we were unsure how to separate the encoder and decoder inputs for the black-box T5 model. Moreover, SimpleT5 did not provide us with the necessary functionalities to compute loss, and we were forced to switch to T5-base for training instead. To solve the issue of the intractability of T5, we decided to use the tokenizer and the loss function that was packaged with T5-base. In this way, both the G2T predictions and log probabilities obtained from model.forward calls are nicely black-boxed.

In terms of model accuracy, our model faced the same challenges as the original model in that we could only update the parameters of one model at a time. For instance, in a T-cycle, we only update the parameters of the G2T model. If we attempted to back-propagate the loss through the initial T2G prediction, the loss would be non-differentiable as T2G post-processing was required to convert the outputs of the T2G prediction into feasible inputs into G2T. Thus, we are forced to mimic the original paper's solution by fixing the T2G parameters and only update the G2T parameters during a T-cycle, and vice versa for a G-cycle.

### 4.4 RESULTS

#### 4.4.1 SUPERVISED T2G

We train the supervised T2G model for twenty epochs with a learning rate of 0.002 with the default Adam optimizer. The mini-batch size in each epoch is 32. The F1 scores were tracked at the end of each epoch using the development dataset. The results of our supervised T2G model are shown below in the table.

| | Micro F1 Score | Macro F1 Score |
|---|---|---|
| OnePass (Wang et al., 2019) | 66.2 | 52.2 |
| Supervised T2G in CycleGT | 60.6 | 50.7 |
| Supervised T2G (Our Implementation) | 71.1 | 60.6 |

Table 1: Supervised T2G Test Results on WebNLG 2017 Data

And the F1 scores after each epoch are shown in the graph:

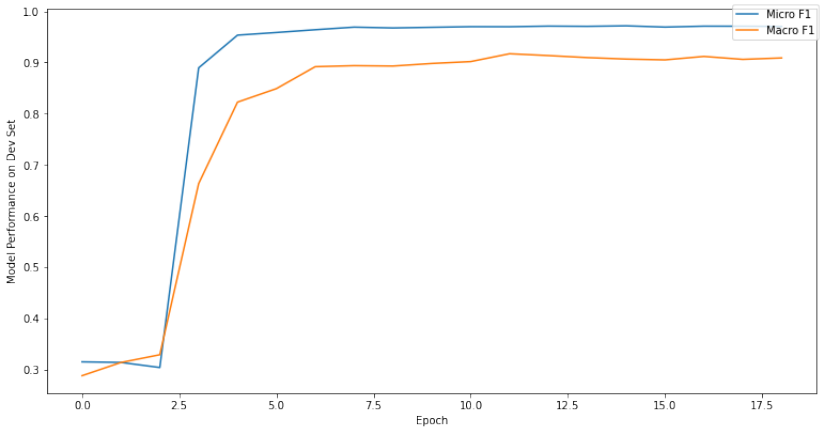


Figure 1: Supervised T2G: F1 Scores on Development Set

#### 4.4.2 SUPERVISED G2T

We implement the supervised G2T model with simpleT5[1], a pre-developed module that supports quick training of T5. We train the model for five epochs with a learning rate of 0.0001 with the AdamW optimizer. The batch size is 8. The results of our supervised G2T model are shown below in the table.

| | BLEU | METEOR | $\text{ROUGE}_\text{L}$ | CIDEr |
|---|---|---|---|---|
| StrongNeural (Moryossef et al., 2019) | 46.5 | 0.392 | 65.4 | 2.87 |
| BestPlan (Moryossef et al., 2019) | 47.4 | 0.391 | 63.1 | 2.69 |
| GraphWriter (Koncel-Kedziorski et al., 2019) | 45.8 | 0.356 | 68.6 | 3.14 |
| Seg&Align (Shen et al., 2020) | 46.1 | 0.398 | 65.4 | 2.64 |
| Supervised G2T in CycleGT | 56.4 | 0.445 | 69.1 | 3.86 |
| Supervised G2T (Our Implementation) | 49.0 | 0.464 | 62.8 | 3.09 |

Table 2: Supervised G2T Test Results on WebNLG 2017 Data

#### 4.4.3 UNSUPERVISED CYCLE TRAINING

The results of our unsupervised cycle training follow.

UNSUPERVISED G2T First, we show the results for our G2T model from unsupervised training:

| | BLEU | METEOR | $\text{ROUGE}_\text{L}$ | CIDEr |
|---|---|---|---|---|
| RuleBased (Schmitt et al., 2020) | 18.3 | 0.336 | - | - |
| GT-BT (Schmitt et al., 2020) | 37.7 | 0.355 | - | - |
| Unsupervised G2T in CycleGT | 55.5 | 0.437 | 68.3 | 3.81 |
| Unsupervised G2T (Our Implementation) | 42.4 | 0.405 | 60.1 | 2.59 |

Table 3: Unsupervised G2T Training Results on WebNLG 2017 Data

UNSUPERVISED T2G Now, we show the results for our T2G model from unsupervised training:

[1] https://github.com/Shivanandroy/simpleT5

| | Micro F1 Score | Macro F1 Score |
|---|---|---|
| RuleBased (Schmitt et al., 2020) | 0.0 | 0.0 |
| GT-BT (Schmitt et al., 2020) | 39.1 | - |
| Unsupervised T2G in CycleGT | 58.4 | 46.4 |
| Unsupervised T2G (Our Implementation) | 72.6 | 52.1 |

Table 4: Unsupervised G2T Training Results on WebNLG 2017 Data

# 5 Analysis

## 5.1 Supervised T2G

Overall, our supervised T2G model did perform better than the other supervised T2G models which were tested on WebNLG 2017 Data, according to the F1-scores, including the supervised T2G model in CycleGT.

One main issue with the T2G model is that it is initially extremely biased towards predicting the $\langle NO\_RELATION \rangle$ tag. This occurs since when we call T2G train on a shuffled dataset, the examples with many entities are shuffled to the front. In these examples, the majority of the entity-entity relation pairs are no relation which induces some bias in the model. To account for this, supervised T2G training is first warmed up with epochs where only examples with less than four entities are used.

We experimented with different learning rates (0.1, 0.002, 0.00001) and batch sizes (8, 16, 32). The best results are obtained with a learning rate of 0.002 and a batch size of 16. There is a drastic difference in F1-scores if we input the entire test dataset as a large batch versus predicting on each example one-by-one as well. This is due to the model performing better on smaller batches, which occurs because less variability in the number of entities occurs in smaller batches. This leads to less predictions being masked and discarded as a result of the necessitated preprocessing.

## 5.2 Supervised G2T

In the original CycleGT paper, they claimed to use T5-Base and had done some parameter tuning to achieve such results on the WEBNLG 2017 dataset. Our implementation only used SimpleT5 with T5-Base and we performed minimal hypertuning of parameters, as our overall goal was to replicate the unsupervised cycle training results. Nevertheless, our implementation still outperformed other methods that did not use T5.

## 5.3 Unsupervised/Cycle Models

Overall, our unsupervised T2G model did perform better than the other supervised T2G models which were tested on WebNLG 2017 Data, based on F1 scores, the unsupervised T2G model in CycleGT.

Our final models use standard learning rates (0.001) and a batch size of 16. However, we also experimented with training the models with smaller learning rates (0.00005 and 0.0002 for T2G and G2T, respectively) and a larger batch size (32).

The graph above shows that these training parameter changes consistently resulted in worse evaluation metrics (on the dev.json dataset) for both G2T and T2G. So, we did not proceed with these models.

Due to limited time and availability of computing resources, we were unable to further tune other hyperparameters and attempt to make other model improvements. However, if we were given the time to do so, we would experiment with learning rate scheduling, activation regularization, embedding dropout, ensembling, and temporal activation in order to further improve our models. We would also like to see how our models perform on the WebNLG 2020 and GenWiki datasets and compare ourselves with other existing models that were evaluated on these datasets.


(a) Unsupervised G2T

(b) Unsupervised T2G

Figure 2: Evaluation Metrics for Different Training Methods

## 6 Conclusion

In our project, we developed both supervised and unsupervised learning methods for T2G and G2T tasks, as well as cycle learning frameworks. In addition to the previous work in CycleGT, we added elements of SimpleT5 in G2T training. For the result, we achieved both higher F1 scores in T2G supervised learning and unsupervised training, and G2T supervised learning with proposed evaluation metrics compared to the original CycleGT paper. Moreover, we explored with unsupervised cycle training and this could be the direction of our further research.

### Author Contributions

Jeffrey Cheng Jeffrey created custom datasets and dataloaders for supervised T2G training, as well as a dataloader for the unsupervised cycle training. He implemented T2G architecture and wrote code for its training and evaluation. Additionally, he worked with Rami in merging T2G and G2T during cycle training and David in debugging the loss calculations for cycle training. Lastly, he helped write some sections of the report, the method, dataset descriptions, and T2G sections.

Rami Sbahi Rami worked with Jeffrey on data processing and writing code for iterative back translation in order to merge the supervised T2G and G2T models into an unsupervised cycle training framework. He also helped write the code for training and evaluating these models, alongside Jeffrey and David. Finally, he wrote several sections of the report, including extensions, model and training details (G2T and iterative back translation), and analysis (unsupervised cycle models).

Ruoyu (David) Wu David performed data processing for the supervised G2T model. He also trained and evaluated the supervised G2T model. He worked on the cycle training framework and especially on implementing the T-cycle. He trained the cycle models together with Rami and debugged loss calculations. Finally, David helped write some sections of the report, including related work, model and training details, and analysis.

Yuncong (Joe) Zuo Joe worked on the supervising training of the T2G models, as well as developing evaluation metrics for T2G supervised and T2G cycle training. In addition to that, Joe was in charge of supervised T2G training, hyperparameter tuning, and model selection. Joe also wrote several sections of the report, including the abstract, introduction, approach, and conclusion.

## REFERENCES

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. The WebNLG challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pp. 124–133, Santiago de Compostela, Spain, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-3518. URL https://aclanthology.org/W17-3518.

Qipeng Guo, Zhijing Jin, Xipeng Qiu, Weinan Zhang, David Wipf, and Zheng Zhang. CycleGT: Unsupervised graph-to-text and text-to-graph generation via cycle training. In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pp. 77–88, Dublin, Ireland (Virtual), 12 2020. Association for Computational Linguistics. URL https://aclanthology.org/2020.webnlg-1.8.

Zhijing Jin, Di Jin, Jonas Mueller, Nicholas Matthews, and Enrico Santus. IMaT: Unsupervised text attribute transfer via iterative matching and translation. 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)*, pp. 3097–3109, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1306. URL https://aclanthology.org/D19-1306.

Keshav Kolluru, Samarth Aggarwal, Vipul Rathore, Mausam, and Soumen Chakrabarti. Imojie: Iterative memory-based joint open information extraction. *CoRR*, abs/2005.08178, 2020. URL https://arxiv.org/abs/2005.08178.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. Text Generation from Knowledge Graphs with Graph Transformers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2284–2293, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1238. URL https://aclanthology.org/N19-1238.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7871–7880, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL https://aclanthology.org/2020.acl-main.703.

Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. A joint neural model for information extraction with global features. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7999–8009, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.713. URL https://aclanthology.org/2020.acl-main.713.

Amit Moryossef, Yoav Goldberg, and Ido Dagan. Step-by-step: Separating planning from realization in neural data-to-text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2267–2277, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1236. URL https://aclanthology.org/N19-1236.

Thien Huu Nguyen and Ralph Grishman. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pp. 39–48, Denver, Colorado, June 2015. Association for Computational Linguistics. doi: 10.3115/v1/W15-1506. URL https://aclanthology.org/W15-1506.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. openAI blog, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Martin Schmitt, Sahand Sharifzadeh, Volker Tresp, and Hinrich Schütze. An unsupervised joint system for text generation from knowledge graphs and semantic parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7117–7130, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.577. URL https://aclanthology.org/2020.emnlp-main.577.

Xiaoyu Shen, Ernie Chang, Hui Su, Cheng Niu, and Dietrich Klakow. Neural data-to-text generation via jointly learning the segmentation and correspondence. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7155–7165, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.641. URL https://aclanthology.org/2020.acl-main.641.

Haoyu Wang, Ming Tan, Mo Yu, Shiyu Chang, Dakuo Wang, Kun Xu, Xiaoxiao Guo, and Saloni Potdar. Extracting multiple-relations in one-pass with pre-trained transformers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1371–1377, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1132. URL https://aclanthology.org/P19-1132.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 207–212, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-2034. URL https://aclanthology.org/P16-2034.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2242–2251, 2017. doi: 10.1109/ICCV.2017.244.