Text-to-Text Pre-Training Model with Plan Selection for RDF-to-Text Generation

Natthawut Kertkeidkachorn **Hiroya Takamura**
Artificial Intelligence Research Center (AIRC)
National Institute of Advanced Industrial Science and Technology (AIST),
2-4-7 Aomi, Koto-ku, Tokyo, 135-0064, Japan
{n.kertkeidkachorn, takamura.hiroya}@aist.go.jp

Abstract

We report our system description for the RDF-to-Text task in English on the WebNLG 2020 Challenge. Our approach consists of two parts: 1) RDF-to-Text Generation Pipeline and 2) Plan Selection. RDF-to-Text Generation Pipeline is built on the state-of-the-art pre-training model, while Plan Selection helps decide the proper plan into the pipeline.

1 Introduction

Natural language generation from data, or data-to-text, aims to generate natural language text that describes the input data such as tables and graphs. WebNLG (Gardent et al., 2017; Ferreira et al., 2018) is one of the data-to-text tasks, where the given input data is a set of triples (a tree or a graph). A triple consists of entities and a relationship between them in the form of (subject, predicate, object) describing one fact. For example, (Donald Trump, birthplace, New York) describes the facts "Donald Trump was born in New York." In the WebNLG task, up to 7 triples are given as the input data. Figure 1 presents an instance of the WebNLG dataset[1], where 3 triples are given as the RDF input graph and the output text is the natural language description describing the fact from triples. Generating text from triples is useful for many applications such as question and answering (He et al., 2017), where a set of triples retrieved as an answer can be organized and translated into natural language text.

Many approaches have been proposed to deal with the WebNLG task. Neural Machine Translation (NMT) approach is one of the popular methods for this task (Gardent et al., 2017). In WebNLG 2017, a neural machine translation-based approach, achieved the highest score in the automatic evaluation (Gardent et al., 2017). In this approach,

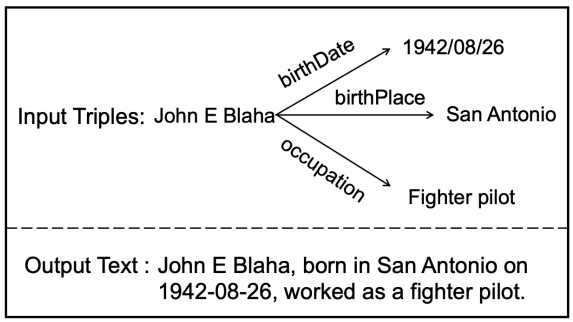

Figure 1: Illustration of the WebNLG 2020 task on RDF-to-text (English).

the delexicalization, where entities are replaced by placeholders, with N-gram search and the linearization using DBpedia type enrichment are used together with the standard encoder-decoder with attention model. Due to the simple linearization, the relationship between entities as the graph in triples might lessen. To preserve such information, a graph-based triple encoder (GTR-LSTM) (Distiawan et al., 2018), is introduced. Later, Moryossef et al. (2019) argued that splitting the generation process into plan selection and text realization could help generate the description text that is faithful to the triples. PlanEnc uses a graph convolution network-based model to predict the order of the triples and then an LSTM with attention and the copy mechanism is employed to generate the text corresponding to the order of triples (Zhao et al., 2020). Recently, Kale (2020) investigated Text-to-Text Transfer Transformer (T5) model for the data-to-text task. With transfer learning ability, the T5 model is the current state-of-the-art for RDF-to-text generation.

To further investigate the T5 model from the study (Kale, 2020), we conduct experiments with the T5 model on WebNLG 2020 challenge [2]. Based

[1] An example is from https://webnlg-challenge.loria.fr

[2] https://webnlg-challenge.loria.fr/challenge_2020/

3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+), Dublin, Ireland (Virtual), 18 December 2020, pages 159–166,

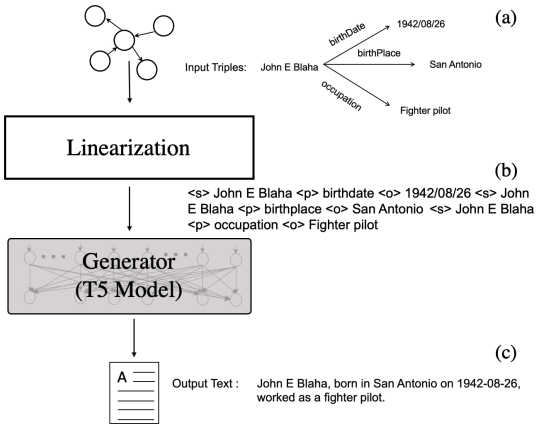

Figure 2: An Overview Pipeline of RDF-to-Text Generation. (a) RDF graph as input, (b) Linearization form of RDF graph and (c) Generated Text.

on our preliminary results, we found that changing the order of the triples before the linearization affected the BLEU score. In this paper, we therefore present a Text-to-text pre-training model for the data-to-text generation with plan selection. In our approach, we follow the study (Kale, 2020), to build the RDF-to-Text Generation pipeline. Then, we develop the graph neural network (GNN) based model to score the plan associating with the triple order. The triple order with the highest plan score is linearized and feed to Generator in the RDF-to-Text Generation pipeline.

2 Approach

In our approach, there are two steps: 1) RDF-to-Text Generation and 2) Plan Selection. RDF-to-Text generation aims to transform input triples into natural language text describing them, while Plan Selection is the pre-processing process to select the order of the triples for the RDF-to-Text Generation pipeline. Note that Plan Selection is performed before RDF-to-Text Generation in the testing phase.

2.1 RDF-to-Text Generation

In the RDF-to-text generation step, we follow a similar approach with the study (Kale, 2020) to build the pipeline. As illustrated in 2, there are two steps in the pipeline of RDF-to-Text Generation: 1) Linearization and 2) Generator.

2.1.1 Linearization

Linearization is to convert RDF graph into the sequence so that the conventional seq-2-seq model can be applied. Specifically, given a set of triples $T = \{(s_1, p_1, o_1), (s_2, p_2, o_2), ..., (s_n, p_n, o_n)\}$, the linearization maps the triples into the word sequence as follows: $< s > s_1 < p > p_1 < o > o_1 < s > s_2 ... < s > s_n < p > p_n < o > o_n$. An example of the linearization is shown in Figure 2 (a) and (b). In Figure 2 (a), the input triples are given as the RDF graph and the linearization form of the graph is converted as in Figure 2 (b). Note that usually in an RDF graph 'Donald Trump' is concatenated into one single token 'Donald_Trump'. In the conversion process, we therefore remove underscore (_) in the entity name to recover the original words denoting the entity.

2.1.2 Generator

Generator is to generate natural language text corresponding to the given word sequence from the linearization step as shown in Figure 2 (b) and (c). In our study, we employ Text-To-Text Transfer Transformer (T5) pre-trained models released by the study (Raffel et al., 2020) as the generator model. The T5 model is built on a transformer-based encoder-decoder architecture. The model was trained through an unsupervised multi-tasking (span masking) in the Colossal Clean Crawled Corpus (C4) dataset as well as through supervised learning tasks including translation, summarization, classification, and question answering. In the training phase, we fine-tune the T5 model by using the linearization form of the RDF graph as the input to target the text description as the output. During the fine-tuning, all parameters in the models are updated.

2.2 Plan Selection

The objective of Plan Selection is to compute scores of linearization forms of triples with different orders of triple and then find the order of triples providing the highest score as the selected plan.

The intuition behind this idea is that permuting the order of triples during the linearization may yield different texts. To concretely elaborate this sensitivity of Generator in the pipeline, we illustrate the example of the permutation of the order of triples in Figure 3 and discuss the intuition using this example. Since there are 3 triples in the example, we can linearize RDF triples into 6 patterns owing to the order of triples. When we input these 6 linearization forms of triples into Generator in the pipeline, Generator returns the different generated texts. For example, with the linearization 1, it

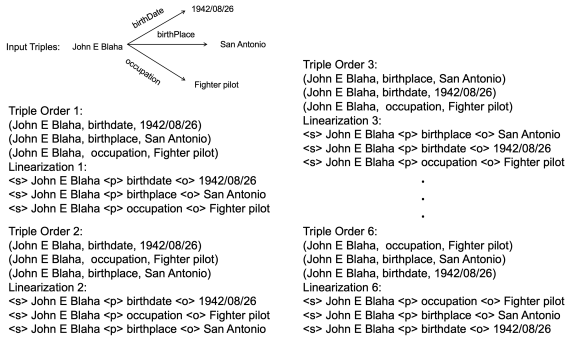

Figure 3: The example of the permuting the triple orders and their Linearization forms.

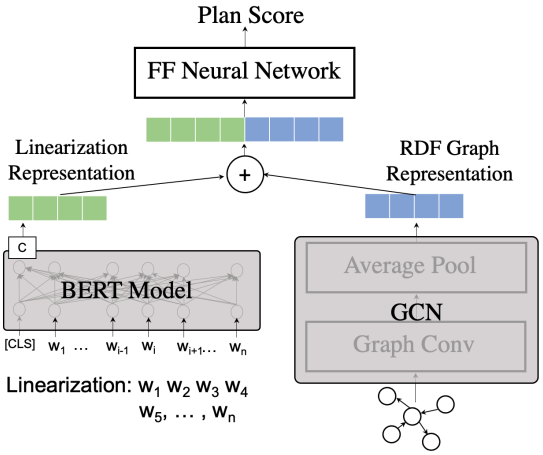

Figure 4: The overview design of Plan Selection.

may generate "*John E Blaha (1942-08-26), born in San Antonio, worked as a fighter pilot*", while with the linearization 3, it may generate "*John E Blaha, born in San Antonio on 1942-08-26, worked as a fighter pilot.*". We then evaluate generated texts and found out that the BLEU score of generated texts on the same RDF graph may differ. When we manually select the order of triples providing the highest BLEU score to examine the upperbound, the BLEU score improved by around 10 points on the development set of WebNLG 2020. Based on this preliminary experiment, we therefore aim to build Plan Selection, which helps select the order of triples for Generator in the RDF-to-Text pipeline.

In Plan Selection, there are 2 steps: 1) Plan Generation and 2) Plan Evaluation.

2.2.1 Plan Generation

Plan Generation is to generate all possible orders of triples given the input triples as plans. In the Plan Generation step, we create all combinations of the triple orders. The concrete example is shown in Figure 3. In the example, we can create 6 orders of triples. Since RDF graphs in WenNLG 2020 Challange are small, exhaustive plan generation is feasible.

2.2.2 Plan Evaluation

Plan Evaluation is to estimate the score of a plan, i.e., an order of triples. We design the Plan Evaluation architecture into three components: 1) Linearization Representation, 2) RDF Graph Representation, and 3) Plan Score as shown in Figure 4. Linearization Representation aims to learn the representation of the Linearization form of triples by the given order. RDF Graph representation is to learn the representation of the RDF graph in the low dimensional vector space. Plan Score is to compute the score for the given triple order corresponding to the RDF graph by using Linearization Representation and RDF Graph Representation.

Linearization Representation is to project the word sequence that linearizes from one order of triples into a low dimensional vector space. In Linear Representation, we learn the representation by feeding the word sequence into the contextualized language representation. Recently, contextualized language representations (Devlin et al., 2019; Peters et al., 2018; Radford et al., 2019; Lewis et al., 2019) gain wide attention from the NLP community. Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019) is one of popular pre-trained models based on the transformer architecture (Vaswani et al., 2017). Due to its performance, it has been used in many NLP tasks. We, therefore, utilize BERT to learn Linearization representations. Given the RDF graph G and the plan p, we can generate the linearization form as the sequence $L = w_1 w_2 w_3 ... w_n$ by mapping the plan to sequence as demonstrated in the example of Figure 3. After obtained the sequence L, we always add a special classification token [CLS] as the beginning of the sequence L. Then, the sequence L with [CLS] token is fed to the pre-trained BERT model. After feeding the input sequence to the pre-train model, the final vector representation C corresponding to [CLS] token is used as the Linearization Representation.

RDF Graph Representation is to embed the entire graph into the low dimensional vector space. In RDF Graph Representation, we first employ Node2Vec (Grover and Leskovec, 2016) to generate the node attributes. We do not represent the node attributes with one-hot vector encoding due

to the sparseness of the RDF graph. Then, we use GraphConv (Morris et al., 2019) with the global average pool to compute the representation of the RDF graph. We formally define the computation in this component as follows. Given the RDF Graph $G = (X, A)$, where X is the matrix representing the node attributes and A is the adjacency matrix of RDF graph, we learn the RDF Graph Representation by the following equation:

$$x_i^{'} = W_1 x_i + \sum_{j \in N(i)} W_2 x_j, \quad (1)$$

where $x_i^{'}$ is the representation of the i-th node, x_i is the attribute vector of the i-th node, x_j is the attribute vector of the j-th node, which is the neighbor node of the i-th node, $N(i)$ is a set of nodes connecting to the i-th node, W_1 and W_2 are the learnable weight matrix trained by propagating the loss in Eq 4. We then use the calculated representations to obtain the RDF representation of graph G:

$$X_G = \frac{1}{|V|} \sum_{v \in V} x_v' \quad (2)$$

where $x_i^{'}$ is the representation of the i-th node and V is the set of the nodes in G.

Plan Score computes the score by using Linearization Representation and RDF Graph Representation. We concatenate Linearization Representation C and RDF Graph Representation X_G into a single vector representation and feed it to the feed-forward neural network (FFNN) as shown in Equation (3). The output of FFNN is the score of the plan subjected to the RDF graph:

$$\hat{y} = [C; X_G] W_{FFNN}^T + b, \quad (3)$$

where W_{FFNN} and b are trainable parameters for FFNN.

In the **training phase**, we need to build the score for each plan as the target for Plan Selection to learn. To obtain the score, we feed all possible linearization of triple orders into the RDF-to-Text generation pipeline. Then, we compute the BLEU score for each order of triples from the corpus. With this strategy, we can get the plan associated with the BLEU score based on the model of the pipeline. We then use the following objective function to optimize all trainable parameters in Plan Selection:

$$\mathcal{L} = \sum_{s \in S} \sum_{p \in P(s)} |y_s - \hat{y}_{s,p}|, \quad (4)$$

where S is a WebNLG 2020 corpus, $P(s)$ is a set of plans for sample s in the corpus.

In the **testing phase**, we compute the scores of all plans for the given RDF graph and then select the plan with the highest score. The triple order associating with that plan is linearized as the word sequence and input to Generator in the pipeline to generate text description.

3 Experiments and Results

3.1 Experimental Setup

The experimental setup is as follows:

Datasets: The dataset used in the experiment is WebNLG 2020 Challenge on the RDF-to-Text (English) task. In the task, there are 35,426, 4,464 and 1,779 RDF Graph-Text pairs for training, validating and testing respectively. RDF Graph-text pairs are generated from 16 distinct DBpedia (Auer et al., 2007) categories: Athlete, Artist, CelestialBody, MeanOfTransportation, Politician, Airport, Astronaut, Building, City, ComicsCharacter, Food, Monument, SportsTeam, University, WrittenWork and Company.

Settings: In our pipeline, we use the pre-trained model t5-base from the huggingface repository[3]. We set the hyperparameters in the fine-tuning process as follows: batch: 8, learning rate: 2.0×10^{-5}, epochs: 20, maximum tokens: 512, and optimizer: Adam (Kingma and Ba, 2014). The default values are used for the other hyperparameters.

For Plan Selection, we set the following hyperparameters: batch: 8, learning rate: 0.001, epoch: 10, and optimizer: Adam. In each component of Plan Selection, we set the hyperparameters as follows. In Linearization representation, we select the bert-base-uncased model as the BERT model with the default setting. In RDF Graph Representation, we set the following hyperparameters for Node2Vec[4]: vector dimension: 128, window size: 10, batch: 32, walk length: 80, walk step: 10 and min count: 1 to create node attributes. We stack GraphConv[5] into 3 layers and use Global Average Pool as the pooling method. In FF Neural Network, we use 2 hidden layers with 410 neurons and the ReLU activation.

Baseline: The baseline in the experiment is provided by WebNLG 2020 Challenge[6].

[3] https://huggingface.co/models

[4] https://github.com/eliorc/node2vec

[5] https://pytorch-geometric.readthedocs.io/en/latest/modules/nn.html#torch_geometric.nn.conv.GraphConv

[6] https://webnlg-challenge.loria.fr/

Table 1: Overall Results of Our Approach on RDF-to-Text of WebNLG 2020 (English)

Metrics	Our Approach		Baseline
	Select	No Select	
BLEU	50.93	50.43	40.57
BLEU_NLTK	0.482	0.476	0.396
METEOR	0.384	0.382	0.373
CHRF++	0.636	0.637	0.621
TER	0.454	0.439	0.517
BERT_P	0.952	0.955	0.946
BERT_R	0.947	0.949	0.941
BERT_F1	0.949	0.951	0.943
BLEURT	0.54	0.57	0.47

Table 2: Results on Seen Category of Our Approach on RDF-to-Text of WebNLG 2020 (English)

Metrics	Our Approach		Baseline
	Select	No Select	
BLEU	60.53	60.99	42.95
BLEU_NLTK	0.522	0.518	0.415
METEOR	0.380	0.381	0.387
CHRF++	0.638	0.641	0.650
TER	0.458	0.432	0.563
BERT_P	0.955	0.961	0.945
BERT_R	0.946	0.948	0.942
BERT_F1	0.950	0.954	0.943
BLEURT	0.51	0.55	0.41

Table 3: Results on Unseen Category of Our Approach on RDF-to-Text of WebNLG 2020 (English)

Metrics	Our Approach		Baseline
	Select	No Select	
BLEU	43.82	43.07	37.56
BLEU_NLTK	0.436	0.430	0.370
METEOR	0.379	0.376	0.357
CHRF++	0.618	0.618	0.584
TER	0.472	0.456	0.510
BERT_P	0.947	0.949	0.944
BERT_R	0.944	0.945	0.936
BERT_F1	0.945	0.946	0.940
BLEURT	0.50	0.54	0.44

Table 4: Results on Unseen Entity of Our Approach on RDF-to-Text of WebNLG 2020 (English)

Metrics	Our Approach		Baseline
	Select	No Select	
BLEU	50.74	49.40	40.22
BLEU_NLTK	0.504	0.490	0.393
METEOR	0.405	0.398	0.384
CHRF++	0.672	0.669	0.648
TER	0.410	0.410	0.476
BERT_P	0.958	0.961	0.949
BERT_R	0.956	0.958	0.950
BERT_F1	0.957	0.959	0.949
BLEURT	0.61	0.64	0.55

Evaluation Settings: There are two evaluation settings: 1) Automatic Evaluation and 2) Human Evaluation.

In Automatic Evaluation, the evaluation metrics are BLEU (Papineni et al., 2002), BLEU_NLTK[7], METEOR (Banerjee and Lavie, 2005), CHRF++ (Popović, 2017), TER (Snover et al., 2006), BERT Precision (BERT_P) (Zhang et al., 2019) , BERT Recall (BERT_R) (Zhang et al., 2019), BERT_F1 (Zhang et al., 2019) and BLEURT (Sellam et al., 2020).

In Human Evaluation, the evaluation metrics are:

- **Data Coverage**: this metric assesses how much information from the data has been covered in the text.
- **Relevance**: this metric assesses whether the text contains any non-presented predicates.
- **Correctness**: this metric assesses whether the text describes predicates with correct objects.
- **Text Structure**: this metric assesses whether the text is grammatical and well-structured, written in good English.
- **Fluency**: this metric assesses the naturalness of generated texts.

3.2 Results

3.2.1 Automatic Evaluation

In Automatic Evaluation, the results are listed in Tables 1-4. These results are provided by WebNLG 2020 (Castro-Ferreira et al., 2020; Moussalem et al., 2020).

challenge_2020/

[7] https://www.nltk.org/_modules/nltk/translate/bleu_score.html

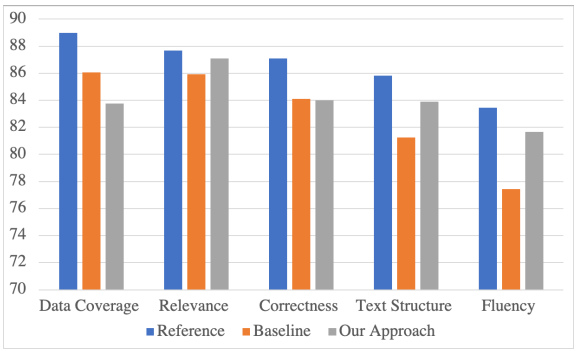

Figure 5: Overall Results of Human Evaluation on RDF-to-Text of WebNLG 2020 (English).

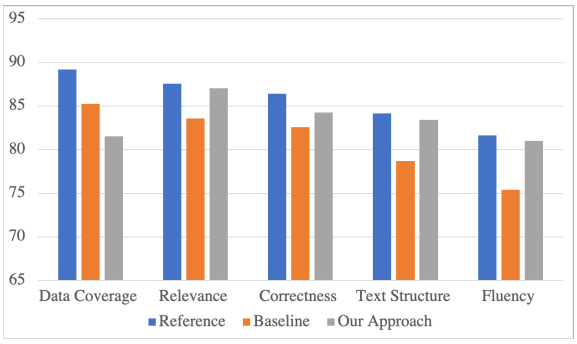

Figure 6: Seen Category Results of Human Evaluation on RDF-to-Text of WebNLG 2020 (English).

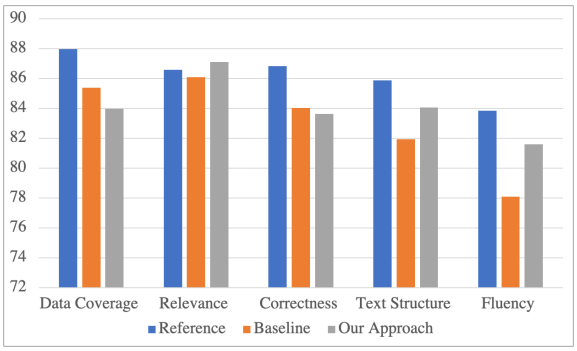

Figure 7: Unseen Category Results of Human Evaluation on RDF-to-Text of WebNLG 2020 (English).

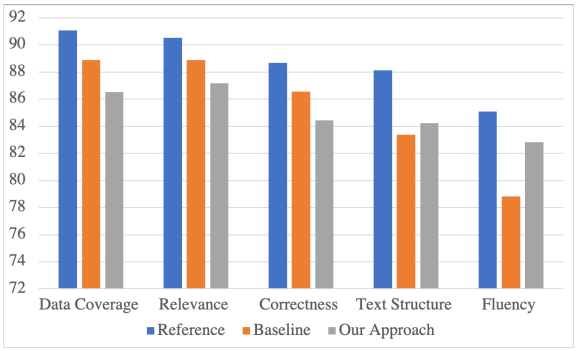

Figure 8: Unseen Entity Results of Human Evaluation on RDF-to-Text of WebNLG 2020 (English).

In our approaches, the pipeline with Plan Selection slightly outperforms the pipeline without Plan Selection on both BLEU score metrics (BLEU and BLEU_NLTK) as shown in Table 1. In the seen category as presented in Table 2, it turns out that our approach with Plan Selection slightly yields better performance than the approach without Plan Selection on only BLEU_NLTK. With this observation, the Plan Selection model therefore does not contribute much to the seen category and even degrades the performance in some cases. Nevertheless, on the unseen category and unseen entity evaluation in Tables3-4, we found our approach with Plan Selection improves both BLEU score metrics when comparing with our approach without Plan Selection.

Based on these results in Tables 2-4, the approach building on the T5 model works well on the seen category and it may not require the plan selection process. Still, on the unseen category and unseen entity, there is a room, where the plan selection could play a role in improving the performances.

So far, we are interested in the BLEU metric because we optimize the Plan Selection model using the BLEU metric. To further investigate the plan selection in this approach, we might optimize the plan selection using other metrics

Overall, our approach with and without the plan selection outperforms the baseline in most metrics except for CHRF++ as shown in Table 1. Although our approach with Plan Selection could slightly improve some metrics, the results do not explicitly surpass our approach without Plan Selection. Therefore, the current contribution of Plan Selection is very limited.

3.2.2 Human Evaluation

In Human Evaluation, the results are illustrated in Figure 5-8. These results are also provided by WebNLG 2020 (Castro-Ferreira et al., 2020; Moussalem et al., 2020). Due to the limitation of the WebNLG 2020 submission, only our approach with Plan Selection is evaluated in the Human Evaluation setting. Note that the reference texts used in the automatic evaluation are also evaluated in this setting.

Overall results in Figure 5 show our approach outperforms the baseline in the relevance metric, the text structure metric, and the fluency metric;

however, the baseline provides the better results in the data coverage metric and the correctness metric. According to the statistical testing, we found statistically significant differences in the data coverage result, the correctness result, and the fluency result. This indicates our approach provides fluency text, while it lacks the coverage and the correctness corresponding to RDF inputs.

To further investigate the results, we analyze the results based on the seen category, the unseen category, and the unseen entity. As shown in Figure 6-8, the data coverage result, the text structure, and the fluency follow the overall result's trend. Nevertheless, the relevance result and the correctness result are varied. It turns out that on the unseen entity our approach could not outperform the baseline in the relevance result; however, our approach achieves even better relevance result than the reference texts in the unseen category. In the seen category, our approach yields higher correctness result than the baseline, while in the unseen category and unseen entity the correctness results follow the overall trend.

In Human Evaluation, we notice that our approach severely suffers from the data coverage as shown in Figure 6-8. The gap between our approach and the reference (even the baseline) is large. This is the limitation of our approach, where we do not force the model to use all inputs.

4 Conclusion

In this paper, we introduce RDF-to-Text Generation with Plan Selection for WebNLG 2020 challenge. The approach comprises two parts: 1) the pipeline for RDF-to-Text Generation and 2) Plan Selection. In the pipeline, we follow the state-of-the-art model for RDF-to-Text Generation (Kale, 2020), while in Plan Selection we propose a method for selecting the order of triples for the pipeline. Overall, we observed a slight improvement in the BLEU score when applying Plan Selection. Nevertheless, Plan Selection only optimized the BLEU score in this study. In the future, we aim to investigate optimize Plan Selection in other metrics.

Acknowledgments

This paper is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO) JPNP20006.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Thiago Castro-Ferreira, Claire Gardent, Nikolai Ilinykh, Chris van der Lee, Simon Mille, Diego Moussalem, and Anastasia Shimorina. 2020. The 2020 bilingual, bi-directional webnlg+ shared task: Overview and evaluation results (webnlg+ 2020). In *Proceedings of the 3rd WebNLG Workshop on Natural Language Generation from the Semantic Web (WebNLG+ 2020)*, Dublin, Ireland (Virtual). Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of NAACL-HLT*, pages 4171–4186.

Bayu Distiawan, 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)*, pages 1627–1637.

Thiago Castro Ferreira, Diego Moussallem, Emiel Krahmer, and Sander Wubben. 2018. Enriching the webnlg corpus. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 171–176.

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*, pages 124–133.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864.

Shizhu He, Cao Liu, Kang Liu, and Jun Zhao. 2017. Generating natural answers by incorporating copying and retrieving mechanisms in sequence-to-sequence learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 199–208.

Mihir Kale. 2020. Text-to-text pre-training for data-to-text tasks. *arXiv preprint arXiv:2005.10433*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. 2019. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4602–4609.

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, Volume 1 (Long and Short Papers)*, pages 2267–2277.

Diego Moussalem, Paramjot Kaur, Thiago Castro-Ferreira, Chris van der Lee, Conrads Felix Shimorina, Anastasia, Michael Röder, René Speck, Claire Gardent, Simon Mille, Nikolai Ilinykh, and Axel-Cyrille Ngonga Ngomo. 2020. A general benchmarking framework for text generation. In *Proceedings of the 3rd WebNLG Workshop on Natural Language Generation from the Semantic Web (WebNLG+ 2020)*, Dublin, Ireland (Virtual). Association for Computational Linguistics.

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 of the Association for Computational Linguistics*, pages 311–318.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Maja Popović. 2017. chrf++: words helping character n-grams. In *Proceedings of the second conference on machine translation*, pages 612–618.

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

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

Thibault Sellam, Dipanjan Das, and Ankur P Parikh. 2020. Bleurt: Learning robust metrics for text generation. *arXiv preprint arXiv:2004.04696*.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, volume 200. Cambridge, MA.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Chao Zhao, Marilyn Walker, and Snigdha Chaturvedi. 2020. Bridging the structural gap between encoding and decoding for data-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, volume 1.