$\mathcal{P}^2$: A Plan-and-Pretrain Approach for Knowledge Graph-to-Text Generation

Qipeng Guo[1,2] **Zhijing Jin**[1] **Ning Dai**[2]
Xipeng Qiu[2] **Xiangyang Xue**[2] **David Wipf**[1] **Zheng Zhang**[1]
[1]Amazon AI [2]Fudan University
qpguo16@fudan.edu.cn

Abstract

Text verbalization of knowledge graphs is an important problem with wide application to natural language generation (NLG) systems. It is challenging because the generated text not only needs to be grammatically correct (fluency), but also has to contain the given structured knowledge input (relevance) and meet some other criteria. We develop a **p**lan-and-**p**retrain approach, $\mathcal{P}^2$, which consists of a relational graph convolutional network (R-GCN) planner and the pretrained sequence-to-sequence (Seq2Seq) model T5. Specifically, the R-GCN planner first generates an order of the knowledge graph triplets, corresponding to the order that they will be mentioned in text, and then T5 produces the surface realization of the given plan. In the WebNLG+ 2020 Challenge, our submission ranked in *1st place* on all automatic and human evaluation criteria of the English RDF-to-text generation task.[1]

1 Introduction

The WebNLG 2020 Challenge (Castro-Ferreira et al., 2020) focuses on automating the conversion between text and knowledge graph domains, both of which are important ways to store and record knowledge. In brief, the challenge is comprised of two tasks and two languages, namely knowledge graph-to-text generation (G2T) and text-to-knowledge graph extraction (T2G), with separate tracks for English and Russian languages. Our team, Amazon AI (Shanghai), participated in both tasks in English and achieved 1st place in each case. This report will primarily focus on our approach for the G2T task (which received by far the most competing submissions); for the other T2G task, we will briefly summarize our simple strategy in Section 5.

[1]Our code is available at `https://github.com/QipengGuo/P2_WebNLG2020`.

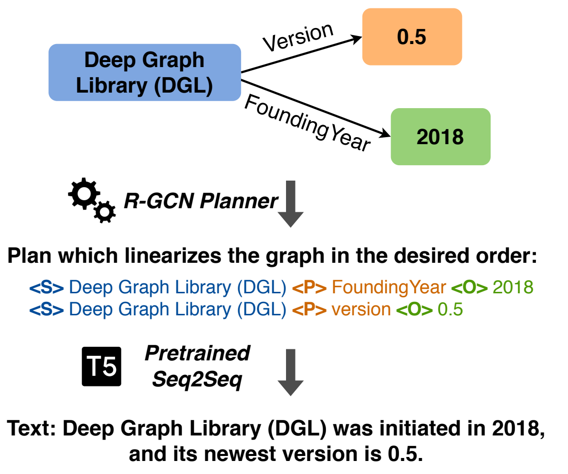
Deep Graph Library (DGL)
Version
0.5
FoundingYear
2018
R-GCN Planner
Plan which linearizes the graph in the desired order:
<S> Deep Graph Library (DGL) <P> FoundingYear <O> 2018
<S> Deep Graph Library (DGL) <P> version <O> 0.5
T5
Pretrained Seq2Seq
Text: Deep Graph Library (DGL) was initiated in 2018, and its newest version is 0.5.

Figure 1: A diagram of our knowledge graph-to-text model. The $\mathcal{P}^2$ model takes in the graph, then (1) linearizes it to a plan with the desired order of triplets by the R-GCN planner, and (2) uses the pretrained T5 model to learn the plan-to-text generation.

For G2T, we introduce a plan-and-pretrain approach, $\mathcal{P}^2$, which uses a text planner based on relational graph convolutional networks (R-GCN) proposed by Zhao et al. (2020), and the pretrained T5 Seq2Seq model (Raffel et al., 2020), as shown in Figure 1. To further boost performance, our $\mathcal{P}^2$ model also incorporates a mechanism to canonicalize the entities in WebNLG with special characters.

Overall, our contributions are as follows:

- We develop a plan-and-pretrain approach, $\mathcal{P}^2$, for knowledge graph-to-text generation. $\mathcal{P}^2$ unleashes the power of the pretrained T5 model by pipelining it with a R-GCN content planner and special canonicalization rules.
- Our $\mathcal{P}^2$ model achieved the *top-1* performance on all criteria of both the automatic and human evaluations of knowledge graph-to-text generation task in English.

- We open-sourced our code and outputs as a reference for future work.

2 WebNLG 2020 Challenge Overview

The English version of WebNLG 2020 has two tasks: The first task aims to automate knowledge graph-to-text generation (G2T), which is an extension of the G2T task in the previous WebNLG Challenge 2017 (Gardent et al., 2017b) with broader categories of knowledge graph and text pairs. Given a knowledge graph of several (subject, predicate, object) triplets, participant systems need to generate verbalizations of the knowledge graph that maintain the same information but in natural language. The other task, text-to-knowledge graph extraction (T2G), involves extracting entities and relations from text to form a knowledge graph.

In terms of data, the English WebNLG 2020 track (Castro-Ferreira et al., 2020) covers knowledge graphs and text from a variety of domains including Airport, Artist, Astronaut, Athlete, Building, Celestial Body, City, Comics Character, Food, Mode of Transportation, Monument, Politician, Sports Team, University, and Written Work.[2] And the test sets includes three additional domains, Film, Scientist, and Musical Work.

Within these data, each knowledge graph consists of 2 to 7 triplets extracted from DBpedia. To collect the text description corresponding to each knowledge graph, crowd-source workers were asked to verbalize the graphs. The statistics of the dataset are shown in Table 1.

Split	Number of Samples
Train (T-G Pairs)	35,426
Valid (T-G Pairs)	4,464
Test (Only G, for G2T)	1,779
Test (Only T, for T2G)	2,155

Table 1: Number of text and knowledge graph pairs in the training, validation, and test sets of the WebNLG 2020 English dataset.

3 $\mathcal{P}^2$ for G2T Generation

Inspired by recent work on the WebNLG 2017 dataset (Ribeiro et al., 2020; Kale, 2020) that achieved state-of-the-art G2T performance with a pretrained Seq2Seq T5 model (Raffel et al., 2020), we also adopt T5 as the backbone of our approach. To produce the input sequence to feed into T5, one direct way following Ribeiro et al. (2020); Kale (2020) is to "linearize" the graph by iterating the (subject, predicate, object) triplets in a random order, using special tokens to specify the subject, predicate, and object of the knowledge triplet. However, feeding the triplets in a random order can introduce burdens for the T5 Seq2Seq model, which must verbalize each triplet while organizing the information within a natural ordering. To assist T5 with the latter, we use a relational graph convolutional network-based planner (R-GCN planner) by Zhao et al. (2020) to learn the best order to linearize a graph.

The overview of our $\mathcal{P}^2$ framework is shown in Figure 1. For a given graph, the R-GCN planner generates a *plan*, which is a sequence of triplets in the form (subject, predicate, object), and the triplets' order corresponds with the order in which they will be expressed in the text. The *plan* is then fed into a Seq2Seq model for the final natural language generation.

Note that the idea of incorporating a planner and a text decoder has been introduced in Zhao et al. (2020), albeit not with a design that is specifically paired with a Seq2Seq model. But given that this work has shown the effectiveness of the planner, we focus on how to integrate it with a pretrained Seq2Seq model in a system. In this regard, our method contains four parts: (1) a R-GCN planner from (Zhao et al., 2020), (2) a rule-based interface that converts plans into a Seq2Seq friendly format, (3) a pretrained T5 Seq2Seq model following the practice of Kale (2020), and (4) some canocalization rules to deal with special characters that are not contained with the dictionary of T5. We now describe each of these components in turn.

3.1 R-GCN Planner

Our R-GCN planner is largely based on (Zhao et al., 2020). The input of the planner is a knowledge graph as represented via the Resource Description Framework (RDF) (Lassila et al., 1998), where the basic unit is the triplet of a subject ($\boldsymbol{s}$), predicate ($\boldsymbol{p}$), and object ($\boldsymbol{o}$), such as (Paris, IsCapitalOf, France). Since there can be multiple triplets (e.g., 2-7 triplets as in the WebNLG dataset), the planner takes in a random order of triplets 1 to N, and aims to find an ideal order of the triplets, such as "$(\boldsymbol{s}_2, \boldsymbol{p}_2, \boldsymbol{o}_2) \rightarrow (\boldsymbol{s}_3, \boldsymbol{p}_3, \boldsymbol{o}_3) \rightarrow (\boldsymbol{s}_1, \boldsymbol{p}_1, \boldsymbol{o}_1)$." The

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

ideal plan we want to generate corresponds to the order that the triplets are mentioned in the ground-truth text.

Following (Zhao et al., 2020), we use an R-GCN (Schlichtkrull et al., 2018) to encode the knowledge graphs. Then, to arrange the triplets in the order that corresponds to the occurrence of their information in the text, the plan is learning in an autoregressive fashion. At each time step, the planner selects the node that represents the triplet that is the most likely to be next, and subsequently we re-run the R-GCN on the updated graph (where the selected triplet's node feature will change).

To train the planner, since it is not straightforward to obtain the order of RDF triplets verbalized in the text from the WebNLG 2020 dataset, we use an outside data source with annotated plans. We use the enriched version of WebNLG 2017, called enriched WebNLG (Ferreira et al., 2018, 2019).[3] It consists of 5,152 training, 644 validation and 1,408 test samples. We only use the training set to train our planner, and reserve the validation set for choosing the best planner model.

3.2 Converting Plans to Sequences

Since the raw output of the planner is an ordered list of triplets, we need to convert the raw plan into a sequential form that is friendly to the T5 as an input. To this end, the beginning of each sequence uses a new task-specific prefix, "Graph to Text:." This serves as the the prefix to all training samples, and also to any subsequent test samples. Additionally, for entities or relations spanning multiple words, we add an underline _ to connect each token in them. For example, the relation type "area code" is changed into "area_code." Each triplet will then be serialized with special tokens signaling the subject, predicate, and object, such as in "Graph to Text: ⟨S⟩ Darlington ⟨P⟩ area code ⟨O⟩ 01325."

And finally, since T5 uses byte-pair encoding, the exact input format of relation type "area code" is actually "area@@ _@@ code" to ensure proper readability. So the final representation of the above example as presented to the T5 model is "Graph to Text: ⟨S⟩ Darlington ⟨P⟩ area@@ _@@ code ⟨O⟩ 01325."

[3]https://github.com/ThiagoCF05/webnlg/tree/master/data/v1.5

3.3 Pretrained Seq2Seq Model for Plan-to-Text Generation

As mentioned previously, to generate text from the plans obtained by the R-GCN planner, we adopt the powerful pretrained model T5 (Raffel et al., 2020), which is the state-of-the-art model (Kale, 2020; Ribeiro et al., 2000) on the WebNLG 2017 dataset (Gardent et al., 2017a), and often seen on other Seq2Seq tasks such as machine translation (Raffel et al., 2020). T5 uses the Transformer architecture (Vaswani et al., 2017) and is pretrained on several large corpora with carefully tuned hyperparameters (Raffel et al., 2020).

The powerful pretraining of T5 has equipped it with strong text generation ability to verbalize the triplets in a fluent way, and also some generalizability that can help on unseen data. For present purposes, we finetune T5 for 10 epochs using the input sequences from Section 3.2.

One potential concern though is whether T5 can cope with the variety of entities in the WebNLG dataset, such as a long airport name "Adolfo Suárez Madrid—Barajas Airport." Such a concern has motivated the previous invention of the copy mechanism (See et al., 2017), which models a switch to select between generating a new word or copying a word from the input text sequence. Despite this potential concern, we find that the T5 can handle the entities in an adept manner, generating complicated entities such as a long phone number without difficulty.

3.4 Canonicalization of Special Tokens

Since the WebNLG data contains lots of special tokens, or non-English characters, e.g., "Adolfo Suárez Madrid—Barajas Airport," we use a special canonicalization algorithm to preprocess the data. Our goal is to convert characters that are not in the English alphabet that T5's byte pair encoding dictionary covers into characters within T5's dictionary. By aligning the dictionary, we can maximize the power of the pretrained T5.

As a representative example, consider the entity "Adolfo Suárez Madrid—Barajas Airport," where "á" is a non-English character that T5 does not recognize, and the long dash "—" is also different from the short hyphen "–" with which T5 is more familiar. For the non-English alphabetical characters, we use an approximate solution, converting them to their English alternatives by the Unidecode

	BLEU	METEOR	chrF++	TER	$\text{BERT}_{\text{Precision}}$	$\text{BERT}_{\text{Recall}}$	BERT_{F1}	BLUERT
ID18 (Ours)	**53.98**	**0.417**	**0.690**	**0.406**	**0.960**	**0.957**	**0.958**	**0.62**
ID30	53.54	0.414	0.688	0.416	0.958	0.955	0.956	0.61
ID30_1	52.07	0.413	0.685	0.444	0.955	0.954	0.954	0.58
ID34	52.67	0.413	0.686	0.423	0.957	0.955	0.956	0.6
ID5	51.74	0.411	0.679	0.435	0.955	0.954	0.954	0.6
ID35	51.59	0.409	0.681	0.431	0.956	0.954	0.954	0.59
ID23	51.74	0.403	0.669	0.417	0.959	0.954	0.956	0.61
ID2	50.34	0.398	0.666	0.435	0.954	0.950	0.951	0.57
ID15	40.73	0.393	0.646	0.511	0.940	0.946	0.943	0.45
ID28	44.56	0.387	0.637	0.479	0.949	0.949	0.948	0.54
Base1	40.22	0.384	0.648	0.476	0.949	0.950	0.949	0.55
Base2	38.07	0.367	0.626	0.515	0.933	0.941	0.932	0.50

Table 2: Performance of top 10 systems and two official baselines (Base1 and Base 2) on the leaderboard of the WebNLG 2020 English RDF-to-text challenge as *ranked by METEOR*. With all metrics, larger is better, with the exception of TER where lower is better. In all categories, our system achieved the first place. Note that ID28 is an unsupervised system (Guo et al., 2020).

package.[4] And for punctuations such as hyphens and quotation marks, the Unidecode package also canonicalizes them to the most standard ones.

Apart from character canocalization, we can also potentially change units of measurement, e.g., "kg/m^3," to the corresponding textual equivalent, e.g., "kilogram per cubic meter," so that the inputs will have more overlap with the corpora that T5 was originally trained on. For the current approach however, we did not actually use this additional preprocessing due to the time limit of the challenge.

After the T5 model generates the output sequence that is canonicalized and does not contain illegal characters, we need to convert it back to its original spelling as provided in the input, otherwise the T5 raw output would score poorly on the test set. (Note that most automatic metrics calculate machine-recognizable word overlaps but not human-intelligible ones.) Consequently, after the T5 raw output, we reverse-engineer the original spelling by coding some rules to infer the original spelling by comparing the raw input data versus its preprocessed version. In this way, our model's final outputs recover the original text form of entities and relations which is consistent with the raw data.

4 Experiments

We first provide implementation details, followed by a presentation of quantitative comparisons using automatic and human evaluation metrics.

4.1 Implementation Details

Beyond the planner module based on Zhao et al. (2020), we adopt the T5-Large model from (Kale, 2020) as our Seq2Seq model. And our implementation is based on DGL (Wang et al., 2019), Pytorch,[5] and Transformers (Wolf et al., 2019). The details of hyperparameters can be found in our open-sourced GitHub repository.

4.2 Automatic Evaluation

For the knowledge graph-to-text generation task, WebNLG 2020 evaluates the text quality by a range of different quantitative metrics (Moussalem et al., 2020) that are listed on an automatic evaluation leaderboard. For this purpose, the quality of textual outputs are assessed using BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), chrF++ (Popovic, 2017), and TER (Snover et al., 2006). In brief, these metrics quantify the n-gram recall, precision, or F-scores between the model outputs and the ground-truth references. Additionally, several BERT-based scores are also reported, including $\text{BERT}_{\text{Precision}}$, $\text{BERT}_{\text{Recall}}$, and BERT_{F1} from Zhang et al. (2020), and BLUERT from Sellam et al. (2020).

We present the performance of the top 10 systems and two official baselines on the leaderboard of WebNLG 2020 English RDF-to-text challenge in Table 2. Our $\mathcal{P}^2$ model achieves the highest out of all systems on all automatic evaluation criteria, indicating higher similarity to the ground truth human written reference in the test set. For example, in terms of BLEU our model outperforms the second system by +0.44, and is over the two official baselines by +13.76 and +15.91, respectively.

[4] https://pypi.org/project/Unidecode/

[5] https://pytorch.org/

	Data Coverage			Relevance			Correctness			Text Structure			Fluency			Avg. Z
	R	Z	P	R	Z	P	R	Z	P	R	Z	P	R	Z	P	
ID18 (Ours)	1	0.222	94.393	1	0.214	95.196	1	0.248	93.531	1	0.305	92.951	1	0.326	90.286	0.263
ID30	1	0.235	95.123	1	0.163	94.615	1	0.224	93.409	1	0.289	92.438	1	0.323	90.066	0.2468
Ref	1	0.251	95.442	1	0.139	94.392	1	0.256	94.149	1	0.254	92.105	1	0.279	89.846	0.2358
ID34	2	0.151	93.169	2	0.117	93.898	1	0.206	92.7	1	0.319	93.089	1	0.327	90.837	0.2240
ID5	2	0.161	93.836	1	0.184	95.22	1	0.224	93.583	1	0.236	91.914	2	0.218	88.688	0.2046
ID23	2	0.116	92.063	1	0.161	94.061	1	0.189	92.053	1	0.258	91.588	2	0.233	88.898	0.1914
ID2	2	0.155	93.291	1	0.164	94.555	1	0.161	91.587	1	0.208	90.752	2	0.185	87.642	0.1746
Base	1	0.17	92.892	1	0.161	93.784	1	0.19	91.794	2	0.039	87.4	3	0.011	82.43	0.1142
ID4	3	-0.075	88.176	1	0.132	92.64	2	0.074	88.626	1	0.168	89.041	2	0.182	86.163	0.0962
ID28	3	0.023	91.231	1	0.125	93.37	2	0.071	89.846	2	0.045	87.879	3	0.072	84.82	0.672
ID15	1	0.259	95.315	1	0.185	94.856	1	0.179	92.489	3	-0.203	83.501	4	-0.161	78.594	0.0518
Base2017	2	0.127	92.066	2	0.113	92.588	2	0.13	90.138	2	-0.064	85.737	4	-0.143	80.941	0.0326
ID12	1	0.272	95.204	1	0.171	94.81	1	0.163	92.128	3	-0.285	81.835	4	-0.241	77.759	0.0160

Table 3: Performance of top 10 systems, two official baselines (Base and Base2017), and the ground-truth reference (Ref) written by human annotators on the leaderboard of the WebNLG 2020 English RDF-to-text challenge. The numbers follow the official website last retrieved in November 2020. On each criteria, we list the rank group (R), the normalized scores (Z), and the raw rating in percentage (P). The table is sorted by the last column, denoted Avg. Z, which indicates the averaged Z values across all five fields.

4.3 Human Evaluation

Since no automatic metric is perfect, the WebNLG 2020 Challenge also compares entries by asking crowdsource workers to evaluate the quality of model outputs. In this regard, human annotators provide ratings with respect to five criteria: (1) data coverage, (2) relevance, (3) correctness, (4) text structure, and (5) fluency. For each criterion, human annotators rate on a 0–100 scale, where 0 is completely disagree and 100 is completely agree. All scores are then normalised (i.e., z-scores). In order to make comparisons statistically significant, all system performance is clustered into groups within which there are no statistically significant differences as quantified by a Wilcoxon rank-sum significant test (Mann and Whitney, 1947).

Table 3 reports results from the leaderboard of human evaluation results on these five criteria. Notably, our model is in the 1st-ranked group on all five criteria.

5 A Simple Baseline for T2G Extraction

We also briefly describe the very simple baseline approach that we submitted for the T2G task at the WebNLG Challenge 2020. Extracting knowledge graph triplets from text contains two steps, the first to identify the entities, and the second to obtain the corresponding relations between them. Accordingly, we first do entity linking to match the entities in the text with the DBpedia ontology, and then query the relation between entities from the DBpedia database.

6 Conclusion

We proposed $\mathcal{P}^2$, a plan-and-pretrain approach that integrates a R-GCN planner and the pretrained T5 model. The resulting performance gain results, at least in part, from unleashing the power of pretrained models, through both the planner and the canocalization of special characters to ensure a consistent vocabulary with T5. The effectiveness of our model is demonstrated through both the automatic and human evaluations of the WebNLG 2020 Challenge, where $\mathcal{P}^2$ achieved a first-place ranking.

References

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: an automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization@ACL 2005, Ann Arbor, Michigan, USA, June 29, 2005*, pages 65–72. Association for Computational Linguistics.

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.

Thiago Castro Ferreira, Chris van der Lee, Emiel van Miltenburg, and Emiel Krahmer. 2019. Neural data-to-text generation: A comparison between pipeline and end-to-end architectures. 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 2019, Hong Kong, China, November 3-7, 2019*, pages 552–562. Association for Computational Linguistics.

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, Tilburg University, The Netherlands, November 5-8, 2018*, pages 171–176. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017a. Creating training corpora for NLG micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 179–188. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017b. The webnlg challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation, INLG 2017, Santiago de Compostela, Spain, September 4-7, 2017*, pages 124–133. Association for Computational Linguistics.

Qipeng Guo, Zhijing Jin, Xipeng Qiu, Weinan Zhang, David Wipf, and Zheng Zhang. 2020. Cyclegt: Unsupervised graph-to-text and text-to-graph generation viacycle training. In *Proceedings of the 3rd WebNLG Workshop on Natural Language Generation from the Semantic Web (WebNLG+ 2020)*, Dublin, Ireland (Virtual). Association for Computational Linguistics.

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

Ora Lassila, Ralph R Swick, et al. 1998. Resource description framework (rdf) model and syntax specification.

Henry B Mann and Donald R Whitney. 1947. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pages 50–60.

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, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318. ACL.

Maja Popovic. 2017. chrf++: words helping character n-grams. In *Proceedings of the Second Conference on Machine Translation, WMT 2017, Copenhagen, Denmark, September 7-8, 2017*, pages 612–618. Association for Computational Linguistics.

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.

António Ribeiro, Gabriel Lopes, and João Mexia. 2000. Using confidence bands for parallel texts alignment. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 432–439, Hong Kong. Association for Computational Linguistics.

Leonardo FR Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. 2020. Investigating pretrained language models for graph-to-text generation. *arXiv preprint arXiv:2007.08426*.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, volume 10843 of *Lecture Notes in Computer Science*, pages 593–607. Springer.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *ACL (1)*, pages 1073–1083. Association for Computational Linguistics.

Thibault Sellam, Dipanjan Das, and Ankur P. Parikh. 2020. BLEURT: learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7881–7892. Association for Computational Linguistics.

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, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.

Minjie Wang, Lingfan Yu, Da Zheng, Quan Gan, Yu Gai, Zihao Ye, Mufei Li, Jinjing Zhou, Qi Huang, Chao Ma, Ziyue Huang, Qipeng Guo, Hao Zhang, Haibin Lin, Junbo Zhao, Jinyang Li, Alexander J. Smola, and Zheng Zhang. 2019. Deep graph library: Towards efficient and scalable deep learning on graphs. *CoRR*, abs/1909.01315.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Chao Zhao, Marilyn A. 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, ACL 2020, Online, July 5-10, 2020*, pages 2481–2491. Association for Computational Linguistics.