Improving Text-to-Text Pre-trained Models for the Graph-to-Text Task

Zixiaofan Yang
Columbia University
zy2231@columbia.edu

Arash Einolghozati
Facebook
arashe@fb.com

Hakan Inan
Facebook
inan@fb.com

Keith Diedrick
Facebook
kdiedrick@fb.com

Angela Fan
Facebook AI Research
angelafan@fb.com

Pinar Donmez
Facebook
pinared@fb.com

Sonal Gupta
Facebook
sonalgupta@fb.com

Abstract

Converting a knowledge graph or sub-graph to natural text is useful when answering questions based on a knowledge base. High-capacity language models pre-trained on large-scale text corpora have recently been shown to be powerful when fine-tuned for the knowledge-graph-to-text (KG-to-text) task. In this paper, we propose two classes of methods to improve such pre-trained models for this task. First, we improve the structure awareness of the model by organizing the input as well as learning optimal ordering via multitask learning. Second, we bridge the domain gap between text-to-text and KG-to-text tasks via a second-phase KG-to-text pre-training on similar datasets and extra lexicalization supervision to make the input more similar to natural text. We demonstrate the efficacy of our methods on the popular WebNLG dataset. Our best model achieves an almost 3 point BLEU improvement on a strong baseline while lowering the relative slot-error-rate by around 35%. We also validate our results via human evaluation.

1 Introduction

There is an abundance of highly accurate knowledge in various forms of structured data organized in tables and knowledge graphs (KGs). Knowledge graphs, such as Wikidata or DBpedia, are essential for many applications, where the data is stored in resource description framework (RDF) format with each triple containing a subject, a property, and an object. Converting a knowledge sub-graph to natural text is an important step for natural communication in question answering applications such as in voice assistants. For example, when the user asks for information about a politician, the system not only needs to retrieve a sub-graph of RDF triples related with that person, but also needs to fluently describe these triples in natural language responses. As such, KG-to-text is a form of Natural Language Generation (NLG) for generating textual descriptions from a sub-graph of a KG.

Traditional approaches separate the KG-to-text task into several micro-tasks, including discourse ordering, sentence structuring, lexicalization, and referring expression generation. However, with the advancement of end-to-end neural network models, attempts have been made to build neural NLG systems that treat the problem as a single task and perform both the planning and realization aspects of text generation simultaneously. Although end-to-end models trained from scratch did not outperform pipeline approaches (Nayak et al., 2017; Moryossef et al., 2019; Ferreira et al., 2019), end-to-end models pre-trained on large-scale text corpora have been shown to achieve state-of-the-art when fine-tuned on the KG-to-text task (Kale, 2020).

However, it is computationally prohibitive to develop large pre-trained models for converting any structure (e.g., KG, Table) to text. Moreover, there is much less available unlabeled structured data than text which is essential for self-supervised pre-training. Thus, we look into ways of adapting the pre-trained text-to-text models for the KG-to-text task. This comes with a few challenges: First, the model is not aware of the implicit planning stage that can be essential in the task to form natural descriptions. In this stage, the triples are reorganized but the content within each triple is kept unchanged to smooth the description of the sub-graph. Second, such language models are optimized for unstructured text and have generally not seen any KG representations during their pre-training phase.

In this work, we study the advantages and limitations of these pre-trained models for the KG-to-text task, and propose improvements from two perspectives. First, we examine improving the model's awareness of structures. This can be achieved via two methods: (1) organizing the input before feeding it to the model; (2) learning the structure in the input by multitask learning of the optimal ordering. Second, we bridge the domain gap between text-to-text and KG-to-text tasks. This is achieved by (3) a second-phase KG-to-text pre-training, as well as (4) extra lexicalization supervision to make the input more similar to natural text.

The rest of this paper is organized as follows: We describe the benchmark dataset and discuss the limitations of text-to-text pre-trained models on KG-to-text task in Section 2. Our proposed improvements are presented in Section 3. The experimental setup and results are discussed in Section 4 and 5. Section 6 describes related work. We conclude in Section 7 and present future directions.

2 Background

2.1 Enriched WebNLG Dataset

The KG-to-text data we use in this work is the enriched version of WebNLG dataset (Ferreira et al., 2018). In the WebNLG dataset (Gardent et al., 2017a,b), there are multiple subject-property-object triples in each entry and multiple text references are provided for each entry that describe the triple set. The triples are structured in diverse graph shapes and the generated text needs to cover the information in all triples. The enriched WebNLG dataset further provides intermediate supervision for each reference text such as gold triple ordering and gold sentence boundary. For example, for an entry with two triples *(William Anders, occupation, Fighter pilot)* and *(William Anders, was a crew member of, Apollo 8)*, the reference text "*Williams Anders was a fighter pilot and a crew member of Apollo 8.*" has a gold triple order of (first, second), and "*William Anders was part of the Apollo 8 crew and was once a fighter pilot.*" has a gold triple order of (second, first). There are 246 distinct properties in the dataset over 10 training domains and 5 unseen test domains. After removing corrupt entries, there are (6940, 872, 1862) entries and (18102, 2268, 4928) reference texts in training, dev, and test sets, respectively.

2.2 Text-to-text Pre-trained Models

In this work, we use BART (Lewis et al., 2019) as our text-to-text pre-trained model as well as our baseline method. Similar to T5 (Raffel et al., 2019) used in Kale (2020), BART is a transformer-based sequence-to-sequence encoder-decoder model pre-trained on unlabeled text data with de-noising objectives, where input is corrupted with noise and the model learns to reconstruct the original text.

Fine-tuning on KG-to-text Fine-tuning text-to-text pre-trained models on the KG-to-text task is a form of transfer learning, which has already been shown useful for other downstream tasks such as document classification, question answering, and summarization (Raffel et al., 2019). With the language model and world knowledge learned from training on large-scale text data, the fine-tuned models generate more natural, fluent text and show better robustness to out-of-domain data. Since BART takes text sequence as input, we represent triples as linearized strings, using tags to indicate subjects, properties, and objects. This is similar to how Kale (2020) used T5. For example, the flattened string of the triple *(Subject, Property, Object)* is "*_s_ Subject _p_ Property _o_ Object*". Multiple flattened triples can be concatenated to form the final input.

KG-to-text Task Challenges The main limitation of fine-tuning BART on KG-to-text, unlike common downstream NLP tasks, is the gap between the expected natural language and the structured RDF triples. This gap is twofold: **(1)** the model is not aware of the implicit planning stage in the KG-to-text task, where the triples are reorganized but the content within is kept unchanged; **(2)** the model also has generally not seen any KG representations during the pre-training phase, making it difficult to understand the triples and grasp the relationships within and between triples.

Table 1 shows examples of typical errors made by the baseline model. In the top example, the second and third triples are incorrectly merged and the fifth triple is completely ignored; in the bottom example, the property in the fourth triple is wrongly described as *preceded by*. These errors are due to insufficient understanding of the implicit triple set structure and the information within triple, as well as poor implicit planning. We also observe that the model is almost perfect on small triple sets, and errors are mostly seen in larger entries with four or more triples while having complex graph structures. This further indicates the limitation of the baseline in understanding triple structures and relations.

3 Improvements upon Pre-trained Language Models for KG-to-text

Inspired by the limitations of the baseline text-to-text pre-trained models for the KG-to-text task, we propose methods to improve the triple structure

Input	Output
(Baked Alaska, country, France) (Hong Kong, leader name, Carrie Lam) (**France**, leader name, Gerard Larcher) (France, language, French language) **(Baked Alaska, region, Hong Kong)**	baked alaska is a dish from france, where the french language is spoken. carrie lam and gerard larcher are leaders in hong kong.
(A Long Long Way, preceded by, Annie Dunne) (A Long Long Way, country, Ireland) (A Long Long Way, publisher, Viking Press) (A Long Long Way, **followed by**, The Secret Scripture)	a long long way was written in ireland and published by viking press. it was preceded by annie dunne and the secret scripture.

Table 1: Typical errors made by pre-trained models fine-tuning on enriched WebNLG dataset.

awareness as well as bridge the gap between the text-to-text pre-training and KG-to-text fine-tuning.

3.1 Improving Triple Structure Awareness

We propose two complementary methods: providing a better organized input triple order to the model, and providing extra supervision to ease the learning of planning.

Better organized input Pre-trained models could be more aware of the relations between the triples if we implicitly or explicitly encoded the tree structure in the flattened input sequence. Direct linearization into a sequence can make learning the tree structure more difficult. For *implicit* structure, we reorder the triples in a fixed tree traversal order, such as breadth-first search (BFS) or depth-first search (DFS). For *explicit* structure, we discard the triple-level input and directly flatten the KG graph into a textual tree representation with brackets (loops are broken randomly).

In addition to ordering triples in tree traversal orders, we also explore having a consistent order for sibling triples that share the same subject. This is inspired by the observation that some properties are often lexicalized before others, for example, the birth date of a person often precedes the death date in description. We find an optimal ordering of the properties which can cover 70% of the sibling property pairs, and use this order for sibling triples.

Multitask Learning of Planning and Text Generation This method is similar to classic NLG models, in which both a planning stage and a realization stage are combined to generate text. As pre-trained models have high capacity, we use the same model to perform planning and KG-to-text jointly in a multitask learning (MTL) fashion: given the input triples, the first task is to *plan* by predicting the indices of the gold triple order and the second task is to generate the text describing the triples.

For the task of planning, we leverage gold triple order, which is extracted from each text reference, indicating the order in which the input triples are described. We consider multiple ways to provide supervision on gold triple order during training in multitask approaches: by concatenating gold order and text in target sequence, by having one encoder and two decoders for two tasks, and by pooling the data of both tasks together and using tags to indicate the corresponding task. We also explore the classic pipeline approach as a cascading alternative of multitask learning, with two pre-trained models, one for predicting triple order and the other for text generation using the predicted order.

3.2 Bridging the Gap between text-to-text and KG-to-text

As mentioned earlier, a major challenge in using pre-trained models for KG-to-text is the gap between natural language and the RDF triples. In this section, we propose two methods to tackle this challenge. The first method brings the text-to-text pre-trained model closer to KG-to-text, while the second brings the KG-to-text data closer to a text-to-text task.

Second-phase Pre-training on a Noisy KG-to-text Dataset Similar to the domain-adaptive pre-training found effective on text classification tasks (Gururangan et al., 2020), the most straightforward approach for bridging the gap between text-to-text and KG-to-text is to perform a second-phase pre-training on data and tasks similar to KG-to-text. We explore utilizing an inverse relation extraction (RE) dataset, DocRED (Yao et al., 2019), containing 96 distinct relations. The goal of RE is to extract triples from text, so the reverse of this task could be roughly treated as KG-to-text. Although additional information might exist in the text compared to in the extracted triples, the noisy second-phase pre-training could still benefit the model's generalization for the final KG-to-text task.

Property Lexicalization In RDF triples, the semantic representations for the properties are often under-specified or even cryptic. Moreover, the property name in the input can be far from its expected surface form. To address this challenge, we curate lexicalization templates for each property in the WebNLG and DocRED datasets. For example, the triple *(Julia Morgan, significant building, Riverside Art Museum)* is lexicalized into "*Julia Morgan designed Riverside Art Museum*". We might also need to change the sentence's voice, e.g., *(British Hong Kong, representative, Chris Patten)* is lexicalized into "*Chris Patten served as representative of British Hong Kong*". By using property lexicalization templates, the triple inputs are converted into a form closer to natural language and are more easily consumed by pre-trained language models.

4 Experimental Setup

Baseline Our baseline model is based on fine-tuning BART using the flattened triples with the original order. We run our experiments five times with different random seeds to have a better idea of the random performance fluctuations. Since small pre-trained models have limited generalization ability after fine-tuning (Kale, 2020), we experiment with BART Base (6 Encoder and Decoder layers, 140M parameters) to find the best configuration and deploy the best setting on BART Large (12 Encoder and Decoder layers, 400M parameters). For pre-processing, we converted camel-case entities into plain text and cleaned special tokens. We fine-tune BART for 50 epochs using Lamb optimizer (You et al., 2019) with learning rate of $1.65e{-4}$ and early stopping with a patience of 10.

Automatic Metrics For measuring similarity between the output and references, we use **BLEU-4** (Papineni et al., 2002) with multiple references as the metric. For semantic fidelity, we use **slot error rate** (SER) measured by the percentage of unique input entities missing in the output. Standard SER only considers exact match of entities in the text output, but many entities in the WebNLG dataset need to be rephrased in the natural text descriptions (e.g., when the property is *nationality*, the object *United States* is often rephrased to *American*). Thus, we modify the SER metric to account for the rephrasing of entities seen in gold references. As such, SER would capture cases where the model is missing some entities (and hence some triples) in the output but would fall short if an entity is misused in other relations. To check whether the improvements on the metrics are statistical significant, we also conduct independent two-sample t-tests with a threshold of 0.05.

		BLEU	SER
	Baseline	55.92	3.96
(a)	+ DFS	**56.96**	**3.83**
	+ BFS	56.25	4.19
	+ DFS w/ P-order	56.14	4.25
	+ BFS w/ P-order	56.02	3.92
(b)	+ Pipeline	55.60	4.25
	+ MTL concat (order‖text)	**56.46**	**2.65**
	+ MTL concat (text‖order)	56.30	3.51
	+ MTL two decoders	**56.47**	3.79
	+ MTL data pooling	55.82	3.30
(c)	+ P-Lex first-lex w/ tag	**58.71**	3.30
	+ P-Lex first-lex w/o tag	58.62	**3.03**
	+ P-Lex merged	58.01	3.87
(d)	+ Pre-train DocRED	56.92	3.71

Table 2: WebNLG test results on using different implementations for (a) triple reordering, (b) multitask learning, (c) property lexicalization, and (d) second-phase pre-training. The abbreviations are provided in the text.

Human Evaluations To measure the occurrence of errors that cannot be captured by automatic metrics, we randomly select entries from test set and annotate the outputs of the baseline and best model for grammaticality and correctness. Specifically, the human annotators are asked two binary questions with customized guidelines: **(1)** Is the generated description grammatical? **(2)** Given the input triple set, is the generated output correct? The latter consists of various aspects such as information omission and hallucination.

5 Results

For each of the improvements proposed in Section 3, we consider multiple alternatives and compare their performance. As shown in Table 2, the test set BLEU of the BART base model is close to the T5 base model (Kale, 2020), and we use this as a strong baseline to study the effectiveness of our proposed techniques.

5.1 Better Organized Input

We observe that explicit tree representation using brackets yields slightly worse results in both BLEU

Baseline	Input	_s_ Antioquia Department _p_ country _o_ Colombia _s_ Bandeja paisa _p_ ingredient _o_ Chorizo _s_ Bandeja paisa _p_ region _o_ Antioquia Department
	Target	Chorizo is an ingredient in Bandeja paisa, a dish from the Antioquia Department region, in Colombia.
DFS	Input	_s_ Bandeja paisa _p_ ingredient _o_ Chorizo _s_ Bandeja paisa _p_ region _o_ Antioquia Department _s_ Antioquia Department _p_ country _o_ Colombia
	Target	same as baseline
MTL concat	Input	same as baseline
	Target	1 2 0. Chorizo is an ingredient in Bandeja paisa, a dish from the Antioquia Department region, in Colombia.
P-Lex w/tag	Input	<p> <s>Antioquia Department</s> is in <o>Colombia</o> </p> <p> <o>Chorizo</o> is ingredient in the dish <s>Bandeja paisa</s> </p> <p> <s>Bandeja paisa</s> is from <o>Antioquia Department</o> </p>
	Target	same as baseline

Table 3: Examples of input and target output after applying the proposed techniques.

and SER when compared with implicit tree traversal methods. This may be due to the even larger gap between the tree representations and natural text. Thus, we only report the results on using implicit tree traversal orders and having a consistent order for sibling triples in Table 2-a.

Between different traversal orders, DFS performs better than BFS, matching our observation that humans tend to describe sub-graphs in a depth-first fashion. However, using a fixed property order for sibling triples (P-order) does not improve the results. In summary, for a better organization of the input, flattening triples in DFS order significantly improves BLEU (p = 0.011) at no extra cost.

5.2 Multitask Learning of Planning and Text Generation

In classic NLG models, text planning and text generation tasks are connected in cascade to form a pipeline. We first follow the pipeline approach using our pre-trained models: one BART is fine-tuned on predicting the indices of the gold triple order, and another BART is fine-tuned on using the predicted triple order from the first BART to generate text. Although the first BART can predict the gold triple order indices with an accuracy of 66% on test set, the output of the second BART is even worse than the baseline model without any planning module (Table 2-b). We contribute the inferior performance of the pipeline model to the error propagation through the two separate stages: for entries with more triples and complex graph structure, the first BART is more likely to predict wrong order, and the inconsistency in the predicted order leads to more error in the generated text.

Next, we focus on the end-to-end approaches and observe that simply concatenating the target sequences of both tasks (order||text) works better than having two decoders or pooling data of the two tasks, with significant improvements in both BLEU (p = 0.039) and SER (p = 0.030) (see Table 2-b). Since planning the order of the triples can be seen as the pre-task of generating text, we hypothesize that the sequential order of concatenation in multitask is the key to the better performance. In other words, during auto-regressive decoding, the decoder could leverage the triple order plan it had previously predicted to help generation, alleviating the problem of missing or modified triples. We conducted another experiment by swapping the concatenated tasks and using (text||order) as the target sequence, and the performance, especially SER, is much worse than using the (order||text) target, in line with our hypothesis.

In order to validate the above hypothesis as to the success of end-to-end multitask models trained on the (order||text) target format, we identified a proxy task setup that is conceptually very similar to our proposed multitask learning: Train an *unshuffling auto-encoder*, where natural language text is broken into contiguous blocks of text and shuffled, and the unshuffling auto-encoder predicts the order of the text blocks as well as the original, unshuffled text. We used the target-side WebNLG dataset for this task, and we computed the exact match between the model output and the unshuffled text on the test set. We found that (order||text) performed substantially better than (text||order) (91.44 vs 84.54), which itself performs slightly better than the no-MTL baseline (82.53), offering strong validation for the hypothesis.

Other alternatives for the planning task are also considered: predicting not only the gold triple order but also the gold sentence boundary of the triples provided in the enriched WebNLG dataset; instead of predicting the indices of gold triple order, output the full flattened triples in gold order; adding triple indices in input sequence to aid the prediction of gold triple indices. However, none of these alternatives yield a significant change in the metrics.

5.3 Property Lexicalization

From Table 2-c, we observed that property lexicalization provides significant improvement on BLEU ($p < 0.001$), due to the higher similarity of flattened triples using the templates and natural text. However, each property might have multiple candidate templates to be lexicalized into. For example, the property *editor* can be lexicalized into either *<subject>'s editor is <object>* or *<object> is the editor of <subject>* depending on voice. We explore different ways of handling properties with multiple templates: **(1)** always use the first template; **(2)** merge all templates when flattening the triple, separated by |; **(3)** use a random template; **(4)** for each possible combination of triple templates, duplicate the entry to form multiple data points.

We observed that (3) and (4) perform significantly worse than others, indicating that a consistent representation for the same property is more important than covering all possible template variations. Moreover, as shown in Table 2-c, (1) *P-Lex first-lex* has higher BLEU and SER than (2) *P-Lex merged*, suggesting that only providing one template for each property is enough for the pre-trained models to understand the semantics of the property, and the model can learn to rephrase the template into other variations as needed.

We also consider two alternatives for flattening the templates: one is using XML-style tags to mark the start and end of the subject, object, and property template; another is to directly insert entities into the template with no tags included to be as close as possible to natural text, similar to Kale and Rastogi (2020). We observed no significant difference between these two alternatives when only applying the property lexicalization technique. However, when combined with multitask learning, the representation with XML-style tags outperforms the one without tags; when combined with second-phase pre-training, the representation without tags outperforms the one with XML-style tags.

5.4 Second-phase Pre-training

The inverse RE dataset that we use for second-phase pre-training is DocRED, which has 3K documents with 96 distinct relations. We segment the documents into sentences, resulting in 17K entries, each containing one sentence and its extracted relations. After the second-phase pre-training, we perform the same fine-tuning process on the WebNLG dataset. The results in Table 2-d show that the further pre-training helps the model adapt to the KG-to-text problem, with a significant increase in BLEU ($p = 0.003$). On the other hand, due to the noisy nature of inverse RE datasets (i.e., existence of extra text not corresponding to any relations), no significant gains on SER is observed.

5.5 Input and Output Format

Table 3 shows examples of flattened input triples and target outputs after applying the best individual configurations of our proposed techniques. For the DFS tree traversal, the order of the flattened triples is modified; for MTL by concatenation, the target output is augmented with the target of the auxiliary planning task; for property lexicalization, the input triples are flattened using templates. Note that for second-phase pre-training, the format of both input and target output is the same as baseline and thus not shown in the table.

5.6 Combination of the Techniques

Here, we study the combined effect of the best individual techniques discussed before. Table 4 shows the automatic evaluation results on the full test set, the domains seen in training set, and the unseen domains in the test set, respectively. We compare the results using the baseline model, the individual techniques, and the combinations of techniques. Instead of exhausting all combinations, we stack the techniques in the order of extra resource used: DFS, MTL by concatenation, property lexicalization, and second-phase pre-training.

Comparing which technique is the most useful By comparing the performance of the individual techniques, we find that property lexicalization results in highest improvement in overall BLEU score, while multitask learning gives the best improvement in overall SER. This trend is also the same for the unseen domains. For seen domains, property lexicalization gives the highest improvement in both metrics. DFS and property lexicalization help the model in both seen and unseen domains, while the improvements of MTL and second-phase pre-training are mostly observed in unseen domains. Note that we also collect lexicalization templates for the unseen properties in the test set, so that for the models involving prop-

	All		Seen		Unseen	
	BLEU	SER	BLEU	SER	BLEU	SER
Baseline	55.92±0.28	3.96±0.68	63.69	1.98	46.53	6.11
+ DFS	56.96	3.83	64.33	1.95	47.99	5.88
+ MTL	56.46	**2.65**	63.41	1.87	47.97	**3.50**
+ P-Lex	**58.71**	3.30	**64.35**	**1.77**	**51.89***	4.97
+ Pre-train	56.92	3.71	63.43	2.2	49.08	5.35
+ DFS + MTL	56.77	2.63	63.28	**1.81**	48.79	3.53
+ DFS + MTL + P-Lex (Best)	**58.67**	**2.48**	63.63	**1.81**	52.65*	**3.21**
+ DFS + MTL + Pre-train	57.42	2.85	63.47	1.95	50.13	3.84
+ DFS + MTL + P-Lex + Pre-train	58.61	2.88	63.49	1.98	**52.67***	3.87

Table 4: Automatic evaluation results of our proposed techniques on the WebNLG test set.

erty lexicalization (marked by * in Table 4), the unseen domains are not strictly "unseen" although the model has not been trained on these domains.

Performance improvement by integrating all techniques When combining the techniques, most of the improvements can be stacked on top of each other. The only exception is when we add second-phase pre-training on a combination of DFS, MTL, and property lexicalition. This might be because the model trained on combining the former three techniques is already very powerful, and the DocRED dataset for KG-to-text second-phase pre-training includes too much noise to be helpful for the model. Taking this into consideration, the best BART base model using our proposed techniques is DFS+MTL+P-Lex, with significant improvements in both BLEU ($p < 0.001$) and SER ($p = 0.012$). We will use this model for qualitative analysis and human evaluations against baseline model in the later sections.

Scaling to larger models To obtain the best performance for text-to-text pre-trained model, we also deploy the best config on BART large model. We achieve 59.97 BLEU and 1.66 SER, compared with the baseline BART large which yields 58.98 BLEU and 3.20 SER. This sets a new state-of-the-art result on the enriched WebNLG dataset.

5.7 Qualitative Analysis

Table 5 shows the generated text of the baseline and the best models. In the top example (seen domain), the baseline model incorrectly merges the first triple into the fifth triple, while the best model understands triple structure better and describes the first triple correctly. The improvements can be more clearly observed in the bottom example (unseen domain), where the baseline model misses the second triple, and incorrectly modifies the property in the first triple, the object in the third triple, and both property and entities in the fourth triple. By using our proposed techniques, the best model not only covers the entities in each triple faithfully, but also lexicalizes the unseen properties better.

5.8 Human Evaluations

We evaluate our best model alongside the baseline model via human evaluations. We randomly select 250 seen and 250 unseen entries, and the outputs of the baseline and the best (DFS+MTL+P-Lex) model on these entries are annotated by 10 human annotators trained on the task with custom correctness and grammaticality guidelines. The results are shown in Table 6. We can see that both models which use BART as the underlying generator enjoy almost perfect grammaticality. On the other hand, our best model's improvement for the correctness is statistically significant which verifies our observations from the previous section. Note that the correctness improvement for the triple sets from unseen domains is larger as expected ($p=0.0125$ in McNemar's test, $N=250$), although we observe average metric improvements with the best model in nearly all evaluations.

6 Related Work

In recent years, most studies on KG-to-text and more broadly data-to-text use neural sequence-to-sequence (seq2seq) systems to generate text. Yet seq2seq systems were found to perform poorly in correctness without proper semantic control (Dušek et al., 2020). Moreover, seq2seq models for text generation tasks treat all input words in a sequential order, often directly linearizing (Konstas et al., 2017; Fan et al., 2019). This linearization is not optimal for encoding data with special structures as in KG-to-text. To address this problem, triple-specific

Input Triples	**(British Hong Kong, representative, Chris Patten)** (William Anders, was a crew member of, Apollo 8) (William Anders, birth place, British Hong Kong) (Apollo 8, crew members, Frank Borman)
Baseline Output	william anders, born in british hong kong, was a crew member of apollo 8 along with frank borman **and chris patten**.
Best Output	william anders, born in british hong kong, served as a crew member of apollo 8 along with frank borman. chris patten is a representative of british hong kong.
Input Triples	(Aston Martin V8, assembly, United Kingdom) **(United Kingdom, capital, London)** (Aston Martin V8, successor, Aston Martin Virage) (Aston Martin Virage, manufacturer, Aston Martin)
Baseline Output	the united kingdom **is the location of the assembly of** aston martin v8, the successor to **aston martin v8**, which was **designed by** the manufacturer **aston martin virage**.
Best Output	aston martin v8 was assembled in the united kingdom, the capital of which is london. it was succeeded by aston martin virage which was created by aston martin.

Table 5: Examples of input and output of baseline and the best BART base model.

	Seen		Unseen	
	C	Gr	C	Gr
Baseline	88.4	98	74	**99.6**
Best	**91.6**	**99.6**	**82.4***	99.2

Table 6: Human evaluation results on correctness (C) and grammaticality (Gr). * indicates $p < 0.02$ in McNemar's test.

or graph-based encoders have been proposed to better process the KG triple input (Vougiouklis et al., 2018; Distiawan et al., 2018; Marcheggiani and Perez-Beltrachini, 2018; Schmitt et al., 2020; Zhao et al., 2020). However, generic pre-trained models cannot be used with a modified encoder and thus the generalizability to unseen domain is limited. There are also works on adding semantic supervisions (Reed et al., 2018; Balakrishnan et al., 2019), however, it requires extensive efforts to annotate discourse relations in each reference text.

Due to the implicit microplanning stage in KG-to-text, there also has been a debate over whether to use a pipeline model or an end-to-end model. Studies found that using separate planning and text generation modules in a pipeline is better than one end-to-end model trained from scratch (Nayak et al., 2017; Moryossef et al., 2019; Ferreira et al., 2019). But with the recent advancement of high-capacity models pre-trained on large-scale text data, end-to-end models achieved state-of-the-art performance without the need of any extra planning module (Kale, 2020). Our work follows this approach while further making the end-to-end pre-trained model better fit the KG-to-text task. Similar with our property lexicalization technique, Kale and Rastogi (2020) explored having natural text templates for each unique slot in dialog acts. However the dialog acts can be treated as a tree with depth of 1, which is easier to understand by a seq2seq model than a KG structure. Our property lexicalization not only considers using templates, but also adds tags to provide structural information. Ribeiro et al. (2020) also used pre-trained models on KG-to-text task and experimented with task-adaptive pre-training. However, in this work BART is around 10 BLEU worse than T5 on WebNLG, but in our experiments BART achieves similar performance, probably indicating significant differences in the fine-tuning settings. Moreover, this work found that pre-trained model with prior world knowledge do not need to understand the KG structure for relatively good performance, in contrast, we focus on improving the KG structure awareness of the model.

7 Conclusions

In this work, we address several challenges of using pre-trained models for KG-to-text. We propose techniques to improve text-to-text pre-trained models for KG-to-text task by addressing triple structure awareness and by bridging the domain gap between text-to-text and KG-to-text. Our techniques boost BLEU and SER over a strong baseline. Our best model yields the SOTA result on the enriched WebNLG dataset and also shows significant improvement over baseline by qualitative analysis and human evaluation. To further utilize the benefit of large-scale pre-training, a future direction is to design a pre-training objective that better fits the KG-to-text task.

References

Anusha Balakrishnan, Jinfeng Rao, Kartikeya Upasani, Michael White, and Rajen Subba. 2019. Constrained decoding for neural nlg from compositional representations in task-oriented dialogue. *arXiv preprint arXiv:1906.07220*.

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.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2020. Evaluating the state-of-the-art of end-to-end natural language generation: The e2e nlg challenge. *Computer Speech & Language*, 59:123–156.

Angela Fan, Claire Gardent, Chloé Braud, and Antoine Bordes. 2019. Using local knowledge graph construction to scale seq2seq models to multi-document inputs. 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)*, pages 4177–4187.

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. *arXiv preprint arXiv:1908.09022*.

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. 2017a. Creating training corpora for NLG micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 179–188, Vancouver, Canada. 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*, pages 124–133.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of ACL*.

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

Mihir Kale and Abhinav Rastogi. 2020. Few-shot natural language generation by rewriting templates. *arXiv preprint arXiv:2004.15006*.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural amr: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157.

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*.

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

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019. Step-by-step: Separating planning from realization in neural data-to-text generation. *arXiv preprint arXiv:1904.03396*.

Neha Nayak, Dilek Hakkani-Tür, Marilyn A Walker, and Larry P Heck. 2017. To plan or not to plan? discourse planning in slot-value informed sequence to sequence models for language generation. In *INTERSPEECH*, pages 3339–3343.

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.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Lena Reed, Shereen Oraby, and Marilyn Walker. 2018. Can neural generators for dialogue learn sentence planning and discourse structuring? In *Proceedings of the 11th International Conference on Natural Language Generation*.

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*.

Martin Schmitt, Leonardo FR Ribeiro, Philipp Dufter, Iryna Gurevych, and Hinrich Schütze. 2020. Modeling graph structure via relative position for better text generation from knowledge graphs. *arXiv preprint arXiv:2006.09242*.

Pavlos Vougiouklis, Hady Elsahar, Lucie-Aimée Kaffee, Christophe Gravier, Frédérique Laforest, Jonathon Hare, and Elena Simperl. 2018. Neural wikipedian: Generating textual summaries from knowledge base triples. *Journal of Web Semantics*, 52:1–15.

Yuan Yao, Deming Ye, Peng Li, Xu Han, Yankai Lin, Zhenghao Liu, Zhiyuan Liu, Lixin Huang, Jie Zhou, and Maosong Sun. 2019. Docred: A large-scale document-level relation extraction dataset. *arXiv preprint arXiv:1906.06127*.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. 2019. Large batch optimization for deep learning: Training bert in 76 minutes. In *International Conference on Learning Representations*.

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.