ReGen: Reinforcement Learning for Text and Knowledge Base Generation using Pretrained Language Models

Pierre L. Dognin
IBM Research
pdognin@us.ibm.com

Inkit Padhi
IBM Research
inkpad@ibm.com

Igor Melnyk
IBM Research
igor.melnyk@ibm.com

Payel Das
IBM Research
daspa@us.ibm.com

Abstract

Automatic construction of relevant Knowledge Bases (KBs) from text, and generation of semantically meaningful text from KBs are both long-standing goals in Machine Learning. In this paper, we present ReGen, a bidirectional generation of text and graph leveraging Reinforcement Learning (RL) to improve performance. Graph linearization enables us to re-frame both tasks as a sequence to sequence generation problem regardless of the generative direction, which in turn allows the use of Reinforcement Learning for sequence training where the model itself is employed as its own critic leading to Self-Critical Sequence Training (SCST). We present an extensive investigation demonstrating that the use of RL via SCST benefits graph and text generation on WebNLG+ 2020 and TEKGEN datasets. Our system provides state-of-the-art results on WebNLG+ 2020 by significantly improving upon published results from the WebNLG 2020+ Challenge for both text-to-graph and graph-to-text generation tasks. More details in https://github.com/IBM/regen.

1 Introduction

Graph representation of knowledge is a powerful tool to capture real-world information where complex relationships between node entities can be efficiently encoded. Automatic generation of Knowledge Bases (KBs) from free-form text and its counterpart of generating semantically relevant text from KBs are both active and challenging research topics.

Recently, there has been an increased interest in leveraging Pretrained Language Models (PLMs) to improve performance for text generation from graph, or graph-to-text (G2T) task (Ribeiro et al., 2020). Indeed, large PLMs like T5 (Raffel et al., 2020) and BART (Lewis et al., 2020) that have been pretrained on vast amount of diverse and variedly structured data, are particularly good candidates for generating natural looking text from graph data.

BART- and T5-related models have been employed by top performers in public challenges such as the WebNLG+ 2020 Challenge (Castro Ferreira et al., 2020b) where both graph-to-text and text-to-graph (T2G) tasks are offered, under the names *RDF-to-Text* and *Text-to-RDF* (semantic parsing) respectively; RDF stands for Resource Description Framework, a standard for describing web resources. One can notice that more teams entered the competition for the G2T task than for T2G as the latter is a much harder task. Best models generally use PLMs and fine-tune them for the target modality at hand (either graph or text). This is possible by re-framing the T2G and G2T generations as a sequence to sequence (Seq2Seq) generation problem, which suits fine-tuning PLMs well. One can therefore hope to leverage the large pretraining of PLMs to improve the overall generation quality.

The Seq2Seq formulation requires any input graph to be linearized as a sequence, which is not unique. This creates an opportunity for data augmentation where multiple linearizations are provided to the model at training time so the model learns the content represented by the graph, not the order of its sequential representation.

In this work, we are interested in leveraging the power of PLMs for both G2T and T2G generation tasks, and will demonstrate the strength of our approach by improving upon the best results of the WebNLG+ 2020 Challenge (rev 3.0) as reported by Castro Ferreira et al. (2020a) for both T2G (Semantic Parsing) and G2T (Data-to-Text) tasks. We will also present results for the TEKGEN Corpus (Agarwal et al., 2021) to show performance on a different, much larger dataset. To illustrate the task of generation, Fig. 1 provides examples of G2T and T2G outputs obtained using the proposed generation framework. The first two sentences of the abstract of this paper were used as input for T2G using our best model. The model generates a graph from the input text by simultaneously extracting

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 1084–1099
November 7–11, 2021. ©2021 Association for Computational Linguistics

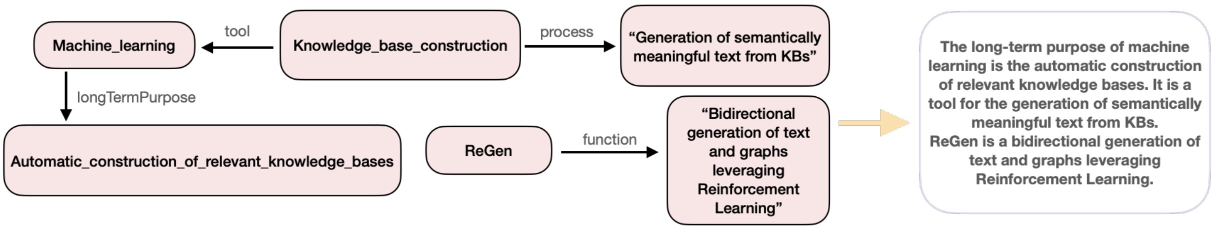
Machine_learning
tool
Knowledge_base_construction
process
"Generation of semantically meaningful text from KBs"
longTermPurpose
Automatic_construction_of_relevant_knowledge_bases
ReGen
function
"Bidirectional generation of text and graphs leveraging Reinforcement Learning"
The long-term purpose of machine learning is the automatic construction of relevant knowledge bases. It is a tool for the generation of semantically meaningful text from KBs. ReGen is a bidirectional generation of text and graphs leveraging Reinforcement Learning.

Figure 1: Actual examples of generation for Text-to-Graph and Graph-to-Text tasks using our best RL models. The first two sentences of the abstract were processed through our best models. First, a graph was created capturing the facts from the input sentences. Then, this graph was used as input to generate text. Despite a strong domain mismatch between input data and models, the generated paragraph is capturing most of the original sentences content. Both models were trained using RL, specifically Self-Critical Sequence Training (SCST).

relevant nodes and linking them coherently. For the G2T task, another model starts from the generated graph and generates semantically relevant text from it. As one can appreciate, the final text is quite readable and captures most facts from the original abstract sentences despite a strong domain mismatch between input data and training data, which both models were built on.

Since both T2G and G2T generative tasks can be formulated as a Seq2Seq problem, we propose to use Reinforcement Learning (RL) as part of the PLMs fine-tuning on the target domain data. For both G2T and T2G tasks, a differentiable function such as the cross-entropy (CE) loss function is often used, since minimizing it results in maximizing the probability of generating the correct token/word. However, when it comes to evaluating a model's performance, benchmarks often use BLEU (Pa Pa Aung et al., 2020), METEOR (Lavie and Agarwal, 2007), and chrF++ (Popović, 2017) for G2T, or simply F1, Precision, and Recall scores for T2G, none of which being differentiable. During training, one hopes that by minimizing the CE loss, the model will tend towards better prediction of the target tokens, hence improving on evaluation metrics as a beneficial by-product. Thankfully, RL provides a framework where we can update our model parameters so to improve evaluation metrics directly. Mixed Incremental Cross-Entropy Reinforce from Ranzato et al. (2016) introduced using REINFORCE (Williams, 1992) for sequence training. We propose to use one of its variant known as Self-Critical Sequence Training (SCST) (Rennie et al., 2017) for both T2G and G2T training.

In summary, our main contributions are:

- We propose to use RL-based sequence training, specifically SCST, for both G2T and T2G tasks. This is the first time that RL based training is proposed to the bi-directional generation of text and graph. To the best of our knowledge, the present work is the first time it is introduced for a T2G task.
- We demonstrate that our approach provides better performance than the best systems reported for the WebNLG 2020+ Challenge.
- We provide a thorough investigation of SCST-based training for both T2G and G2T tasks, including best rewards combination.
- We constructed subject and relation-object boundaries from TEKGEN sentence-triples pairs and showed performance of our approach for both T2G and G2T tasks.
- We adapted the large-scale TEKGEN corpus (Agarwal et al., 2021) for T2G and G2T tasks and confirmed the benefit of SCST-based fine-tuning approach over CE-trained baselines.

2 Related work

In the WebNLG+ 2020 Challenge, most top performing models relied on fine-tuning of PLMs. Interestingly, all four top teams in this Challenge proposed quite different approaches while leveraging PLMs. 1st place Amazon AI (Guo et al., 2020a) pipelined a relational graph convolutional network (R-GCN) and a T5 PLM with some canonicalization rules. 2nd place OSU Neural NLG (Li et al., 2020), the closest to our approach in spirit, used T5 and mBART PLMs to fine-tune after special data preprocessing. 3rd place FBConvAI (Yang et al., 2020) used BART PLM and multiple strategies to model input RDFs. 4th place bt5 employed a T5 PLM trained in a bi-lingual approach on English and Russian, even using WMT English/Russian parallel corpus.

Recently, Dognin et al. (2020); Guo et al. (2020b, 2021) proposed models trained to generate in both T2G and G2T directions, with consistency cycles created to enable the use of unsupervised datasets.

In contrast, our approach of fine-tuning a T5 PLM is fully supervised but can produce either the specialized models for T2G and G2T tasks alone, or a hybrid model that can handle both T/G inputs simultaneously to generate the corresponding translated G/T outputs.

Note that in contrast to many WebNLG+ 2020 Challenge participants, e.g. Li et al. (2020), no preprocessing of the data is performed for text, while for graph triples, we add tokens to mark subject, predicate, and object positions in their linearized sequence representation. Moreover, data augmentation is performed by allowing random shuffling of triples order in graph linearization to avoid a model to learn the exact order of triples, especially for the T2G task.

While the use of RL training in PLM has been explored in many works, the approach of Chen et al. (2020) is closest to ours. However, their work focuses on the improved text generation in the context of natural question generation, while in our algorithm we use it for graph-to-text and text-to-graph generations.

3 Models

Models are trained on a dataset $\mathcal{D}$ composed of a set of $(x_{\text{T}}, x_{\text{G}})^i$ samples, where superscript i denotes the i-th sample in $\mathcal{D}$, x_{T} is made of text (one or more sentences), and x_{G} is a corresponding graph represented as a list of triples $x_{\text{G}} = [(s^1, p^1, o^1), \ldots, (s^K, p^K, o^K)]$, where the k-th triple is composed of a subject s^k, predicate (relationship) p^k, and object o^k. For G2T, the model is given x_{G} as input and must generate $\hat{x}_{\text{T}}$. A cross-entropy loss is computed as an expectation:

$$\mathcal{L}^{\text{T}}_{\text{CE}} = \mathop{\mathbb{E}}_{x_{\text{T}}\sim\mathcal{D}}\left[-\log p_\theta^{\text{G2T}}(x_{\text{T}})\right], \qquad (1)$$

where $p_\theta^{\text{G2T}}(x_{\text{T}})$ is the distribution of the generated sequence $\hat{x}_{\text{T}} = T_{\text{G2T}}(x_{\text{G}})$, $T_{\text{G2T}}(.)$ being the transformation from graph to text. Our model is parameterized by θ, and x_{T} is effectively sampled from the marginal distribution of text samples from $\mathcal{D}$. $\hat{x}_{\text{T}} = [\hat{w}_1, \hat{w}_2, \ldots, \hat{w}_T]$ is a sequence of generated tokens/words. Similarly, for training a T2G model, the cross-entropy loss used in training is simply

$$\mathcal{L}^{\text{G}}_{\text{CE}} = \mathop{\mathbb{E}}_{x_{\text{G}}\sim\mathcal{D}}\left[-\log p_\theta^{\text{T2G}}(x_{\text{G}})\right], \qquad (2)$$

where $p_\theta^{\text{T2G}}(x_{\text{G}})$ is the distribution of the generated graph $\hat{x}_{\text{G}} = T_{\text{T2G}}(x_{\text{T}})$, $T_{\text{T2G}}(.)$ being the transformation from text to graph, and where x_{G} is drawn from the marginal distribution of graph samples from $\mathcal{D}$.

In both Eq. (1) and Eq. (2), x_{G} must be expressed as a sequence of tokens t_j such that a list of triples x_{G} turns into a list of tokens $[t_1, t_2, \cdots, t_M]$. This is simply done by adding tokens marking the subject, predicate, and object boundaries in the sequence such that each triple (s^k, p^k, o^k) is turned into a sequence such as $[\texttt{<S>}, w_1^s, \texttt{<P>}, w_1^p, w_2^p, \texttt{<O>}, w_1^o, w_2^o, w_3^o]$, assuming our subject is made of 1 token, our predicate of 2 tokens, and our object of 3 tokens in this example. <S>,<P>, and <O> are just special marker tokens to help the model know where subject, predicate and objects are located in the sequence.

We start from a pretrained encoder-decoder $\mathcal{M}$ model that we fine-tune on either T2G to get $\mathcal{M}_{\text{T}}$, or G2T task to get $\mathcal{M}_{\text{G}}$. We also propose a third kind of model $\mathcal{M}_{\text{T+G}}$ to be fine-tuned on *both* T2G and G2T samples, i.e. the model will learn to generate in any direction, by supplying an input sample $x = [x_{\text{T}}; x_{\text{G}}]^\top$ and corresponding target for it. Input from each modality is prefixed by a task specific string to distinguish transfer directions ("Text to Graph:" for x_{T} and "Graph to Text:" for x_{G}). For $\mathcal{M}_{\text{T+G}}$ models, the cross-entropy loss is similarly defined as for Eq. (1) and Eq. (2) such that $\mathcal{L}^{\text{T+G}}_{\text{CE}} = \mathop{\mathbb{E}}_{x\sim\mathcal{D}}\left[-\log p_\theta(x)\right]$. All models are shown in Fig. 2. By convention, we refer to models in this paper by their input modality T, G, or T+G.

3.1 Reinforcement Learning

Sequence generation can be seen as an agent making sequential decisions of picking words from a given vocabulary. The agent reacts to its environment by accounting for past predictions and getting rewarded along the way, while its state is defined by the partial sequence generated so far. This interpretation enables the reformulation of Seq2Seq generation within the Reinforcement Learning (RL) framework (Sutton and Barto, 2018; Silver, 2015). More precisely, a sequence generation task can be recast as a Markov Decision Process (MDP) where the agent behavior follows a policy $\pi(a_t|s_t)$. Action a_t corresponds to picking a particular word w_t at time t from a vocabulary $\mathcal{V}$, conditioned on state s_t expressed as the partial sequence generation $s_t = \hat{x}_{1:t} = [\hat{w}_1, \ldots, \hat{w}_t]$, that is sequence of words/tokens already picked. $\pi(a_t|s_t)$ is a stochastic policy that defines a probability distribution of a_t. Once the action a_t is taken,

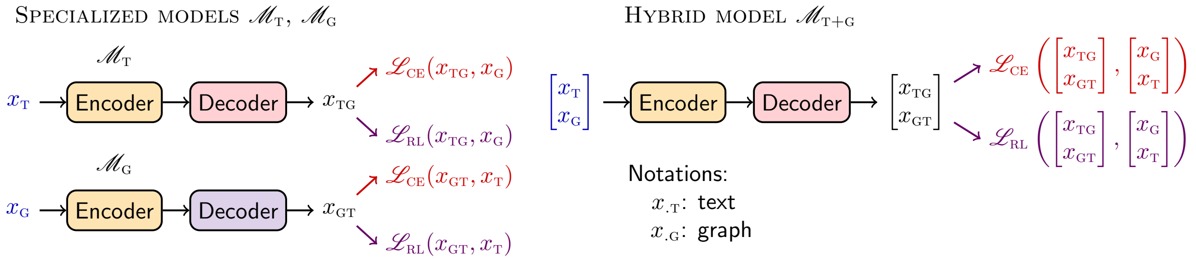
SPECIALIZED MODELS $\mathscr{M}_{\mathrm{T}}$, $\mathscr{M}_{\mathrm{G}}$
$\mathscr{M}_{\mathrm{T}}$
x_{T}
Encoder
Decoder
x_{TG}
$\mathscr{L}_{\mathrm{CE}}(x_{\mathrm{TG}}, x_{\mathrm{G}})$
$\mathscr{L}_{\mathrm{RL}}(x_{\mathrm{TG}}, x_{\mathrm{G}})$
$\mathscr{M}_{\mathrm{G}}$
x_{G}
Encoder
Decoder
x_{GT}
$\mathscr{L}_{\mathrm{CE}}(x_{\mathrm{GT}}, x_{\mathrm{T}})$
$\mathscr{L}_{\mathrm{RL}}(x_{\mathrm{GT}}, x_{\mathrm{T}})$
HYBRID MODEL $\mathscr{M}_{\mathrm{T+G}}$
$\begin{bmatrix} x_{\mathrm{T}} \\ x_{\mathrm{G}} \end{bmatrix}$
Encoder
Decoder
$\begin{bmatrix} x_{\mathrm{TG}} \\ x_{\mathrm{GT}} \end{bmatrix}$
$\mathscr{L}_{\mathrm{CE}}\left(\begin{bmatrix} x_{\mathrm{TG}} \\ x_{\mathrm{GT}} \end{bmatrix}, \begin{bmatrix} x_{\mathrm{G}} \\ x_{\mathrm{T}} \end{bmatrix}\right)$
$\mathscr{L}_{\mathrm{RL}}\left(\begin{bmatrix} x_{\mathrm{TG}} \\ x_{\mathrm{GT}} \end{bmatrix}, \begin{bmatrix} x_{\mathrm{G}} \\ x_{\mathrm{T}} \end{bmatrix}\right)$
Notations:
$x_{.\mathrm{T}}$: text
$x_{.\mathrm{G}}$: graph

Figure 2: Specialized and hybrid models rely on the same losses for fine-tuning. However, specialized models are dedicated to a particular generation task while hybrid models can handle both generation directions.

the agent receives a reward $r_t = r(s_t, a_t)$ before it transitions to the next state s_{t+1}. A sequence of actions $a_{1:T} = [a_1, \ldots, a_T]$ is selected until the end of generation is reached. The agent aims at maximizing the expectation of cumulative reward

$$J(\pi) = \mathbb{E}_\tau \left[\sum_{t=1}^{T} \gamma^t r_t \right] = \mathbb{E}_\tau \left[R(\tau) \right] \qquad (3)$$

where γ is a discounting factor used to control the horizon of the cumulative reward, $\gamma \in [0, 1]$. The expectation is taken over trajectories τ, sequences made of $\{s_1, a_1, r_1, \ldots, s_T, a_T, r_T\}$, where a_t was chosen from policy $\pi(a_t|s_t)$. RL provides both *on-policy* and *off-policy* approaches to maximize $J(\pi)$ in Eq. (3). We are particularly interested in *on-policy* techniques that rely on data samples generated from the model to train, especially since our models start from large fine-tuned PLMs that can already generate good samples. This helps avoid the common drawback of on-policy techniques of generating poor samples at first when trained from scratch. These policy-based (Williams, 1992; Zaremba and Sutskever, 2016) and actor-critic based techniques (Bahdanau et al., 2017; Rennie et al., 2017) have been studied for text generation and often update the underlying model with policy gradient (Ranzato et al., 2016; Li et al., 2016; Tan et al., 2019; Paulus et al., 2017). Policy-based methods focus on a parameterized policy π_θ where θ is optimized to maximize $J(\pi_\theta)$. The policy $\pi_\theta(a_t|s_t)$ is the PLM generative model p_θ, CE fine-tuned as described at the beginning of Section 3.

REINFORCE, presented by Williams (1992), allows the optimization of a model's parameters θ by maximizing the expected value of the word-based reward $R_w(\hat{x}_{\mathrm{T}})$ of generated sequence $\hat{x}_{\mathrm{T}} = [\hat{w}_1, \ldots, \hat{w}_T]$. For notation convenience, note that $R_w(\hat{x}_{\mathrm{T}}) = R(\tau)$ since we are now dealing with sequence of words/tokens $\hat{x}_{\mathrm{T}}$ selected by the actions in trajectory τ. We will also use the $R(\hat{x}_{\mathrm{T}})$ notation for simplicity. In order to match common Deep Learning conventions, we can minimize a loss expressed as the negative value of the expected cumulative reward:

$$\begin{aligned} \mathcal{L}_{\mathrm{RL}} &= - \sum_{[\hat{w}_1, \ldots, \hat{w}_T]} p_\theta(\hat{w}_1, \ldots, \hat{w}_T) R_w(\hat{w}_1, \ldots, \hat{w}_T) \\ &= - \mathbb{E}_{[\hat{w}_1, \ldots, \hat{w}_T] \sim p_\theta} R_w(\hat{w}_1, \ldots, \hat{w}_T), \\ &= - \mathbb{E}_{\hat{x}_{\mathrm{T}} \sim p_\theta} R_w(\hat{x}_{\mathrm{T}}). \qquad (4) \end{aligned}$$

$R_w(\hat{x}_{\mathrm{T}})$ is the reward for the generated text which is often associated with non-differentiable metrics such as BLEU, METEOR, chrF, etc. Note that in sequence generation, these metrics-based rewards are available only once a *whole* sequence is generated, trading sparsity/delay of reward for quality (i.e. we use the full sequence reward, not an estimation of partial future reward). We circumvent the non-differentiability issue by using the REINFORCE policy gradient method:

$$\nabla_\theta \mathcal{L}_{\mathrm{RL}} \propto - \left(R(\hat{x}_{\mathrm{T}}) - b\right) \nabla_\theta \log p_\theta(\hat{x}_{\mathrm{T}}), \qquad (5)$$

where b is a baseline used to reduce the variance of our gradient estimate. b can be any function, even a random variable, as long as it is independent of the actions taken to generate $\hat{x}_{\mathrm{T}}$, as described in Chapter 13.4 from Sutton and Barto (2018). In Self-Critical Sequence Training (SCST) (Rennie et al., 2017), b is chosen to be the reward of x^*_{T}, the output generated by the model by greedy max generation, hence the model serving as its own critic:

$$\nabla_\theta \mathcal{L}_{\mathrm{SCST}} \propto - \left(R(\hat{x}_{\mathrm{T}}) - R(x^*_{\mathrm{T}})\right) \nabla_\theta \log p_\theta(\hat{x}_{\mathrm{T}}), \qquad (6)$$

where $\hat{x}_{\mathrm{T}}$ is sampled from our model and x^*_{T} is generated by greedy max. An interesting property of the baseline is that if $R(\hat{x}_{\mathrm{T}}) > R(x^*_{\mathrm{T}})$, sampled $\hat{x}_{\mathrm{T}}$ has higher reward than x^*_{T}, then the model is updated to reinforce the choices made by this generation. In the opposite case where $R(\hat{x}_{\mathrm{T}}) < R(x^*_{\mathrm{T}})$, the model update will take the negative gradient to subdue such generation. When $R(\hat{x}_{\text{T}}) = R(x^*_{\text{T}})$, no update is performed on the model since the gradient is effectively zeroed out, regardless of the individual values $R(\hat{x}_{\text{T}})$ and $R(x^*_{\text{T}})$. This happens when $\hat{x}_{\text{T}}$ and x^*_{T} are identical (greedy-max and sampled sequences are the same). In that case the sample is lost for RL as no update to the model will result from this sample. Basically, REINFORCE is a Monte Carlo method of learning where a gradient update is applied in the direction decided by how $R(\hat{x}_{\text{T}})$ compares to baseline b, the role of b being to reduce the variance of the gradient estimate. Variations around REINFORCE exist on how to apply the gradients, such as MIXER from Ranzato et al. (2016), or on how to evaluate the baseline (Luo, 2020) to minimize the gradient variance.

In our training, PLMs are first fine-tuned using $\mathcal{L}_{\text{CE}}$ loss. Once they reach a good generation quality, the training is switched to RL fine-tuning by minimizing $\mathcal{L}_{\text{SCST}}$.

4 Experimental Setup

In this Section, we present the experimental setup used for all the results reported in this paper.

Models We used T5 PLMs from Wolf et al. (2020) for our experiments for two distinct models, *t5-large* (770M parameters) and *t5-base* (220M parameters), with a special focus on t5-large as it is the best performing of the two on various NLP tasks. Models were fine-tuned to be either specialized on T2G ($\mathcal{M}_{\text{T}}$) or G2T ($\mathcal{M}_{\text{G}}$) task, or to accommodate both directions of generation ($\mathcal{M}_{\text{T+G}}$).

Data processing Graphs are often represented as list of triples. However our model expects a sequence of input words/tokens to work on. The linearization of graph triples is obviously ambiguous as there are many ways to traverse a graph (Breadth First Search, Depth First Search, random walk, etc.). In practice, we linearize the triples in the order of the list provided by the dataset, but use this inherent linearization ambiguity as an opportunity to do data-augmentation. Indeed, models are first fine-tuned using cross-entropy loss that strongly penalizes generation if it is in any different order than the ground truth order. To avoid the model to overfit to our data and memorize observed triples order, we augment the data by including a few permutations of the graph triples.

During graph linearization, we encode the subject, predicate, and object positions by using <S>,<P>,<O> tokens. In practice, we expand the model vocabulary with these special indivisible tokens that are not split during tokenization. No other preprocessing is done on the data for training. We explored masked and span-masked LM fine-tuning to match T5 pretraining (Raffel et al., 2020) which did not lead to any noticeable improvements.

4.1 Datasets

WebNLG+ 2020 We report results on WebNLG+ 2020 (v3.0) used in the WebNLG 2020 Challenge (Castro Ferreira et al., 2020b). The Challenge comprises of two tasks: RDF-to-text generation (G2T), and Text-to-RDF semantic parsing (T2G). The Resource Description Framework (RDF) language is used to encode DBpedia and is commonly used in linked data framework. WebNLG+ uses RDF to encode graphs as sets of triples which are associated to one or more lexicalizations of one or more sentences each. Data for English and Russian are provided, but we only worked on the English subset made of 13,211 train, 1,667 dev, 2,155 testA (semantic parsing), and 1,779 testB (data-to-text) samples (triples sets w/ lexicalizations). The data is clustered semantically into 16 categories *seen* in train and dev sets (Airport, Astronaut, Building, etc.), while 3 categories (Film, Scientist, and Musical-Work) were introduced in test and are *unseen*, i.e. not present in training; see Castro Ferreira et al. (2020a) for more details. Results are aggregated for *all*, *seen*, and *unseen* categories during evaluation. Note that in the literature, prior works sometimes report 'WebNLG' results on previous dataset version, with completely different performance ranges. We compare all our results to WebNLG+ 2020 (v3.0) numbers reported by Castro Ferreira et al. (2020a) in their Table 6 for G2T, and Table 10 for T2G tasks, using the provided official scoring scripts.

TEKGEN To further study the robustness of our system, we also provide experiments using TEKGEN dataset recently introduced in Agarwal et al. (2021). The graph-sentence alignments are curated using Wikipedia and Wikidata. This serves as a perfect large scale test-bed for both G2T and T2G tasks. Unfortunately, this dataset lacks in entity/relation/object boundaries, which makes it difficult to evaluate systems for T2G tasks. In order to address this issue, we further process the triple-text (with no triple boundaries) to create list of triples using Wikidata properties lookup, via Wikidata

WebNLG G2T Team/model	BLEU↑	BLEU↑ NLTK	METEOR↑	chrF++↑
Amazon AI (Shanghai) (Guo et al., 2020a)	0.540	0.535	0.417	0.690
OSU Neural NLG (Li et al., 2020)	0.535	0.532	0.414	0.688
FBConvAI (Yang et al., 2020)	0.527	0.523	0.413	0.686
bt5 (Agarwal et al., 2020)	0.517	0.517	0.411	0.679
ReGen (Ours) G2T.CE t5-large	0.553	0.549	0.418	0.694
ReGen (Ours) G2T.RL t5-large	**0.563**	**0.559**	**0.425**	**0.706**
ReGen (Ours) G2T.CE.ES t5-base (early CE)	0.522	0.518	0.404	0.675
ReGen (Ours) G2T.RL.ES t5-base (early CE)	0.531	0.527	0.410	0.686
ReGen (Ours) G2T.CE.best t5-base (best CE)	0.524	0.520	0.404	0.677
ReGen (Ours) G2T.RL.best t5-base (best CE)	0.527	0.523	0.408	0.681

Table 1: G2T Best results on WebNLG 2020 Challenge (v3.0) dataset. The first four rows were the top performers of the Challenge. Results for CE and RL models are presented for our ReGen systems so to show gains from using SCST. Our G2T.RL is the best system overall, fine-tuning a t5-large model using METEOR reward. G2T.RL.ES and G2T.RL.best show the impact of using early stopping (ES) or best CE selection for starting SCST fine-tuning on a t5-base smaller model while using BLEU_NLTK reward.

Query Service. Additionally, we limit the validation set and test set to 5K and 50K sentence-triples pairs respectively. Our training split after processing contains 6.3 million sentence-triples pairs. As a contribution to the work, we will present the steps to augment TEKGEN dataset with appropriate subject, object and relation boundaries, which enables conventional evaluation of research systems. An example of the processed TEKGEN is shown in Fig. 3 in Appendix.

Metrics WebNLG+ 2020 provides automatic metrics to evaluate models. For G2T, we used BLEU, BLEU_NLTK, METEOR, and chrF++ that are provided by the challenge. For T2G, F1, Precision, and Recall scores are utilized and computed for 4 levels of match: Exact, Ent_Type, Partial and Strict as described in Castro Ferreira et al. (2020a), which loosely correspond to different levels of relaxation of how close a match of an entity must be to the ground truth in content and position in a triple. Note that when generating graphs/RDFs, scoring metrics explore all possible permutations of a graph edges. For TEKGEN, we use the same metrics as for WebNLG+ 2020.

5 Results

For all experiments, PLMs were first exposed to the target datasets (WebNLG+, TEKGEN) by fine-tuning using $\mathcal{L}_{\text{CE}}$ loss. They were then switched to RL training by optimizing the $\mathcal{L}_{\text{SCST}}$ loss. Although no exact recipe has been established for Seq2Seq RL-training, starting from a good CE model helps RL training performance in practice (Ranzato et al., 2016; Rennie et al., 2017). Therefore, we followed the subsequent simple approach: During fine-tuning, the evaluations are conducted on the validation set. From the CE phase, the best performing model iteration is selected based on the METEOR and F1 score for the G2T and T2G tasks, respectively, to pursue RL fine-tuning. In case of G2T, potential ties in METEOR scores among candidate models, are resolved by using BLEU_NLTK, followed by the chrF++ metric. Note that early stopping selection of CE models led to good performance for t5-base models as well. During the SCST phase, the best model iteration on the validation set is selected and its performance numbers on the test set are reported in our tables.

WebNLG+ 2020 G2T For the WebNLG+ 2020 Challenge, the results of the top four systems for RDF-to-text task can be found in Tab. 1 for all categories (results for seen and unseen categories are given in Tab. 5 in the Appendix), while descriptions the top teams' systems were given in Section 2. We report our G2T results for both t5-large and t5-base models as well. For t5-large, ReGen G2T.CE is the best model from CE fine-tuning. ReGen G2T.RL is best model performance for SCST training while using METEOR as reward when starting from G2T.CE model. Tab. 1 shows that our CE model is better than models from all top teams, and the SCST results further improve significantly in

WebNLG T2G Team/model	Match	F1↑	Precision↑	Recall↑
Amazon AI (Shanghai) (Guo et al., 2020a)	Exact	0.689	0.689	0.690
	Ent_Type	0.700	0.699	0.701
	Partial	0.696	0.696	0.698
	Strict	0.686	0.686	0.687
bt5 (Agarwal et al., 2020)	Exact	0.682	0.670	0.701
	Ent_Type	0.737	0.721	0.762
	Partial	0.713	0.700	0.736
	Strict	0.675	0.663	0.695
ReGen (Ours) T2G.CE	Exact	**0.723**	**0.714**	**0.738**
	Ent_Type	**0.807**	**0.791**	**0.835**
	Partial	**0.767**	**0.755**	**0.788**
	Strict	**0.720**	**0.713**	**0.735**
ReGen (Ours) T2G.RL	Exact	0.720	0.712	0.734
	Ent_Type	0.804	0.789	0.829
	Partial	0.764	0.752	0.784
	Strict	0.717	0.709	0.731

Table 2: T2G Best results on WebNLG+ 2020 (v3.0) dataset. The top two teams were the first and second place winner of the Challeneg. Our T2G.CE model improves upon all metrics for all matching schemes, providing a new state-of-the-art results for this Challenge task. T2G.RL models, while still better than previous best results, does not improve upon its CE counterpart.

TekGen G2T Model		BLEU↑	BLEU↑ NLTK	METEOR↑	chrF++↑
ReGen-CE	Val	0.240	0.241	0.231	0.400
	Test	0.241	0.242	0.233	0.405
ReGen-SCST	Val	0.258	0.259	0.240	0.418
	Test	**0.262**	**0.262**	**0.242**	**0.422**

Table 3: G2T Results for TekGen dataset. ReGen-CE establishes a baseline on this dataset. ReGen-SCST consistently improve on the baseline on all metrics, for validation and test sets.

all metrics achieving state-of-the-art results to our knowledge. The gain obtained by SCST alone is quite significant and demonstrates the benefits of RL fine-tuning for this task. We report our best model results in Tab. 1, as well as mean and standard deviation results for multiple random number generator seeds in Tab. 10 in Appendix. When averaging results for few seeded models, sustained gains from SCST are seen for all metrics.

Multiple reward candidates were investigated (BLEU, BLEU_NLTK, METEOR, chrF) as well as some linear combinations of pairs of them, as can be seen in Tab. 7 in Appendix. In Tab. 7, for t5-large, METEOR is consistently the best SCST reward, and improves all the other metrics scores as well. However, for 'smaller' models such as t5-base, BLEU_NLTK is revealed to be the best reward for improving BLEU performance as expected. Again, SCST brings significant gains across all the metrics in that case. Note that for t5-base model, selecting a METEOR reward improves METEOR results significantly as reported in Tab. 9 in Appendix.

Another interesting fact is that early stopping of CE model G2T.CE.ES (at 5 epochs) leads to the best SCST model G2T.RL.ES for t5-base, while selecting the best CE model G2T.CE.best (at 11 epochs) still showed some gains from SCST model G2T.RL.best. SCST needs a good starting point, but a better CE model that has seen a lot more epochs of our dataset maybe harder for SCST to stir in a better solution in the parameter space.

Moreover, the test split contains unseen categories not present in the validation dataset which render choices based on validation sub-optimal for the test dataset. The best models we report in this work are specialized models $\mathcal{M}_{\mathrm{G}}$. Early in our investigation, hybrid models were the best performing model for G2T reaching 0.547 BLEU, 0.543 BLEU_NLTK and 0.417 METEOR, and first to beat the Challenge winning team. However, when batch size became larger (20-24 samples), the specialized models took the lead and retain it still.

For training, we optimized all our models using AdamW (Loshchilov and Hutter, 2017), variant of the Adam optimizer with default values of $\beta = [0.9, 0.999]$ and weight decay of 10^{-2}. For learning rate, we used 5.10^{-6} for all our experiments as it was better than 10^{-5} and 10^{-6} as seen in Tab. 8 in Appendix. All our models were trained with 20-24 minibatch size on WebNLG. Further details on our experimental setup are provided in the Appendix in Section A.

WebNLG+ 2020 T2G Results for the Text-to-RDF task are reported in Tab. 2 for all categories. Results for our best model on seen and unseen categories are given in Tab. 6 in Appendix. Amazon AI and bt5 are the top performing teams. Again, the proposed ReGen T2G.CE model shows strong results that are better in term of *all* metrics, for *all* matching categories. In themselves, these numbers are a de-facto new state-of-the-art for this dataset, as far as we know. SCST model T2G.RL fails to improve on this model though. The *exact F1* metric was used as reward, but the model could never pull ahead of the CE model in our experiments. The exact F1 metric may not be a strong enough reward to really capture the dynamics of graph generation properly for WebNLG+ as it is very rigid in its measure (one must have an exact match), although the same reward gave good results on our second dataset TekGen. A more sensitive metric could possibly help. We even tried to use n-gram based metrics (like BLEU) but to no avail. We further address this issue at the end on this Section.

TekGen G2T For the TekGen dataset, we present our results on Graph-to-Text generation in Tab. 3. Similar to the experiments in WebNLG+, we pick the best model during the CE fine-tuning based on the METEOR score and proceed with the RL fine-tuning. We observe that the RL fine-tuning step helps boost the test split scores on all metrics. It is worth noting that the scores are slightly underestimating the potential of our system because of the nature of the sentences in the TekGen dataset. Unlike WebNLG+, in a paired text-graph sample in TekGen, the linearized graph does not usually cover all the concepts described in the corresponding text. This leads to underestimating when the hypothesis is scored against the reference using n-gram metrics.

T2G Model		F1↑	P↑	R↑
ReGen-CE	Val	0.622	0.608	0.647
	Test	0.619	0.605	0.643
ReGen-SCST	Val	0.615	0.600	0.640
	Test	**0.623**	**0.610**	**0.647**

Table 4: T2G TekGen Results: ReGen-CE establishes a baseline of the dataset. ReGen-SCST improves results on the test set compared to ReGen-CE.

TekGen T2G Results for the Text-to-Graph for TekGen are reported in Tab. 4. Once the CE fine-tuning is done, we continue with the RL fine-tuning using exact F1 as reward. The performance is consistent with what we observe in G2T task for TekGen, where SCST step boosts the performance of the model. Since, we reformulate this dataset (refer Section 4.1) to offer as T2G and G2T tasks, our approach is the first attempt in understanding the nature of TekGen dataset and our methods provide a baseline for future research. Please note that for both T2G and G2T tasks in TekGen, we only start a t5-large PLM.

Summary Results on WebNLG+ 2020 and TekGen demonstrated that RL fine-tuning of models leads to significant improvements of results for T2G and G2T, establishing new state-of-the-art results for both tasks. For WebNLG+, T2G was a challenging task for RL fine-tuning. In further work, we plan to address this issue by investigating two points: First, look into a more sensible graph-dependent sampling for graph structures, rather than the current multinomial sampling of the best tokens at each generation step. Second, try a different reward schemes where the reward is more attuned to the challenges of graph generation as well as graph structure, allowing for some curriculum learning, or increasing the harshness of rewards gradually during training. Results on TekGen showed that RL fine-tuning is a viable option even on large-scale datasets. To enrich this quantitative study of ReGen, we provide a few qualitative cherry picked results in Tab. 11 and Tab. 12 in Appendix.

6 Conclusions

In this paper, we proposed to use RL for improving upon current generation for text-to-graph and graph-to-text tasks for the WebNLG+ 2020 Challenge dataset using pre-trained LMs. We not only defined a novel Seq2Seq training of models in T2G and G2T generation tasks, but we established state-of-the-art results for WebNLG+ for both tasks, significantly improving on the previously published results. We provided extensive analyses of our results and of the steps taken to reach these improvements. We then expanded our approach to large scale training by means of TEKGEN where we demonstrated that RL fine-tuning provides a robust way to improve upon regular model fine-tuning within a dataset that is orders of magnitude larger than the WebNLG+ starting point. We established gains despite a weaker content overlap in text-graph data pairs for TEKGEN. Along the way, we constructed subject, and relation-object boundaries from TEKGEN sentence-triples pairs that we plan on releasing to benefit the research community.

Future work will focus on developing a variant of SCST that leverages the unique structure of graph by either performing of more sensible graph-dependent sampling, or by investigating different reward schemes more attuned to integrating the content and structure of graphs.

7 Broader Impact Statement

The techniques proposed in this paper are inherently dependent on the training data and the PLMs used for fine-tuning on this data. The models do benefit from the large amount of data seen by the PLM they are derived from, however it is fair to assume that any detectable bias in the original data or PLMs would most likely be transferred to the text-to-graph and graph-to-text generative models. This is something to keep in mind when building these generative models. Public datasets were used for all experiments. The TEKGEN with recreated boundaries does not change the underlying data and should not add any further noise nor bias to the original data.

References

Oshin Agarwal, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. 2021. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3554–3565, Online. Association for Computational Linguistics.

Oshin Agarwal, Mihir Kale, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. 2020. Machine translation aided bilingual data-to-text generation and semantic parsing. In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 125–130, Dublin, Ireland (Virtual). Association for Computational Linguistics.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Thiago Castro Ferreira, Claire Gardent, Nikolai Ilinykh, Chris van der Lee, Simon Mille, Diego Moussallem, and Anastasia Shimorina. 2020a. The 2020 bilingual, bi-directional WebNLG+ shared task: Overview and evaluation results (WebNLG+ 2020). In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 55–76, Dublin, Ireland (Virtual). Association for Computational Linguistics.

Thiago Castro Ferreira, Claire Gardent, Nikolai Ilinykh, Chris van der Lee, Simon Mille, Diego Moussallem, and Anastasia Shimorina, editors. 2020b. *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*. Association for Computational Linguistics, Dublin, Ireland (Virtual).

Yu Chen, Lingfei Wu, and Mohammed J. Zaki. 2020. Reinforcement learning based graph-to-sequence model for natural question generation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Pierre Dognin, Igor Melnyk, Inkit Padhi, Cicero Nogueira dos Santos, and Payel Das. 2020. DualTKB: A Dual Learning Bridge between Text and Knowledge Base. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8605–8616, Online. Association for Computational Linguistics.

Qipeng Guo, Zhijing Jin, Ning Dai, Xipeng Qiu, Xiangyang Xue, David Wipf, and Zheng Zhang. 2020a. $\surd^2$: A plan-and-pretrain approach for knowledge graph-to-text generation. In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 100–106, Dublin, Ireland (Virtual). Association for Computational Linguistics.

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

Qipeng Guo, Zhijing Jin, Ziyu Wang, Xipeng Qiu, Weinan Zhang, Jun Zhu, Zheng Zhang, and David Wipf. 2021. Fork or fail: Cycle-consistent training with many-to-one mappings. In *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, volume 130 of *Proceedings of Machine Learning Research*, pages 1828–1836. PMLR.

Alon Lavie and Abhaya Agarwal. 2007. METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231, Prague, Czech Republic. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016. Deep reinforcement learning for dialogue generation.

Xintong Li, Aleksandre Maskharashvili, Symon Jory Stevens-Guille, and Michael White. 2020. Leveraging large pretrained models for WebNLG 2020. In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 117–124, Dublin, Ireland (Virtual). Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2017. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101.

Ruotian Luo. 2020. A better variant of self-critical sequence training.

San Pa Pa Aung, Win Pa Pa, and Tin Lay Nwe. 2020. Automatic Myanmar image captioning using CNN and LSTM-based language model. In *Proceedings of the 1st Joint Workshop on Spoken Language Technologies for Under-resourced languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL)*, pages 139–143, Marseille, France. European Language Resources association.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization.

Maja Popović. 2017. chrF++: words helping character n-grams. In *Proceedings of the Second Conference on Machine Translation*, pages 612–618, Copenhagen, Denmark. 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.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 1179–1195. IEEE Computer Society.

Leonardo F. R. Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. 2020. Investigating pretrained language models for graph-to-text generation.

David Silver. 2015. Lectures on reinforcement learning. URL: `https://www.davidsilver.uk/teaching/`.

Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA.

Bowen Tan, Zhiting Hu, Zichao Yang, Ruslan Salakhutdinov, and Eric Xing. 2019. Connecting the dots between mle and rl for sequence prediction.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi 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 Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Zixiaofan Yang, Arash Einolghozati, Hakan Inan, Keith Diedrick, Angela Fan, Pinar Donmez, and Sonal Gupta. 2020. Improving text-to-text pretrained models for the graph-to-text task. In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 107–116, Dublin, Ireland (Virtual). Association for Computational Linguistics.

Wojciech Zaremba and Ilya Sutskever. 2016. Reinforcement learning neural turing machines - revised.

A Training Setup

All our experiments were run using NVIDIA V100 GPUs for training and validation, some trainings were done on A100. We distributed our training to 2-4 GPUs depending on availability. Each training epoch for CE ranged from 30 minutes to 1 hour depending on number of GPUs utilized.

Validation and testing (1,779 and 2,155 samples for testA and testB of WebNLG+ 2020) lasted from 40 minutes to 1 hour depending on machines. Computation was dominated by beam search generation as we used beam search with beam size of 5 and a max sequence length of 192 (since linearized graph sequence can be quite long). We used the official scoring scripts released by WebNLG+ 2020 Challenge to score all our experiments. The evaluation of graph being the most computationally expensive as all possible matching combinations are tested in what looks like a factorial complexity, taking scoring of set of triples larger than 8 from impractical to not feasible.

All our models were built using PyTorch. Total effective batch sizes were set to either 20 or 24 samples for our distributed training. We adjusted the batch size on each worker to ensure consistent global batch size of 20 or 24.

We did some search on learning rates for t5-large training and SCST rewards, see discussion and results in Section C.

All our trainings have a seeded random number generator for reproducibility. We also report results on WebNLG+ 2020 G2T tasks for *each* training setup by showing results for 3 models from different seeds, and provide means and standard deviations of these results in Tab. 10.

B WebNLG+ 2020 Results per Categories for Best G2T and T2G Models

In Tab. 5, we are reporting results for all WebNLG+ 2020 categories for our best CE and RL models. While results for unseen categories are much worse than for seen categories, RL fine-tuning manages to improve on both seen and unseen categories.

Tab. 6 provides results for seen, unseen and all categories for our best CE model ReGen T2G.CE which established state-of-the-art results on T2G task of WebNLG+ 2020 Challenge dataset.

C Ablation Studies

In Tables 7 and 8 we present ablation studies of different optimized metrics and learning rates for SCST training. As can be seen from Table 7, when METEOR is used as a reward, we get the best performance across all the metrics. We also tried using a combination of multiple rewards with different scaling but did not get any gain over the single metric rewards. In Table 8. we also show the effect of learning rate on SCST performance. Using $lr = 5 \cdot 10^{-6}$ gave us the best performance, while higher rates, such as 10^{-4}, led to unstable training and collapse of SCST.

D G2T Results t5-base models for SCST with METEOR Reward

Results for SCST fine-tuning of t5-base models using a METEOR reward are compiled in Tab. 9. Clearly, these models achieve better METEOR results as expected since they are RL optimized on this metric.

E G2T Results for Models from Multiple Random Seeds

All our training have a seeded random number generator for reproducibility. We also report the mean and standard deviations for all our G2T models. Each model setup was run 3 times using three independent and distinct seeds, following the same exact process. This is to ensure that our results are not just the product of a lucky system configuration or otherwise advantageous random shuffling of our training dataset. All results are reported in Tab. 10.

The gain reported between CE and RL for our t5-large models are clearly still showing after average of all 3 models from distinct random seeds. For t5-base, gains between CE and RL are still present, albeit smaller than for our best systems.

F Generation Examples for G2T and T2G

We present some cherry-picked examples for G2T in Tab. 12 and for T2G in Tab. 11 for both WebNLG and TEKGEN datasets.

G Processed TEKGEN Dataset

In Fig. 3 we show an example of our processing of TEKGEN dataset in establishing subject, relation, object boundaries. This enables both training and evaluating systems for T2G and G2T tasks.

WebNLG G2T Best Models	Category	BLEU↑	BLEU↑ NLTK	METEOR↑	chrF++↑
Ours t5-large ReGen-CE	unseen	48.76	0.489	0.397	0.653
	seen	59.73	0.592	0.433	0.722
	all	55.26	0.549	0.418	0.694
Ours t5-large ReGen-SCST	unseen	49.06	0.493	0.404	0.665
	seen	61.22	0.605	0.440	0.734
	all	56.25	0.559	0.425	0.706

Table 5: G2T: Results for seen, unseen, and all categories subsets in WebNLG+ 2020 Challenge Test dataset. As expected, unseen categories much worse results than for seen categories. RL fine-tuning manages to improve on both seen and unseen categories.

WebNLG T2G ReGen T2G.CE	Match	F1↑	Precision↑	Recall↑
unseen	Exact	0.5809	0.5662	0.6069
	Ent_Type	0.7014	0.6741	0.7497
	Partial	0.6453	0.6241	0.6826
	Strict	0.5754	0.5608	0.6012
seen	Exact	0.8322	0.8286	0.8384
	Ent_Type	0.8878	0.8811	0.8998
	Partial	0.8604	0.8553	0.8696
	Strict	0.8317	0.8282	0.8379
all	Exact	0.7229	0.7144	0.7376
	Ent_Type	0.8067	0.7910	0.8345
	Partial	0.7668	0.7547	0.7882
	Strict	0.7202	0.7118	0.7349

Table 6: T2G: Results for seen, unseen, and all categories subsets in WebNLG+ 2020 Challenge Test dataset. As expected the performance drops significantly for unseen categories and are the best for seen categories.

SCST Reward	BLEU↑	BLEU↑ NLTK	METEOR↑	chrF++↑
BLEU	0.556	0.552	0.420	0.698
BLEU NLTK	0.558	0.554	0.422	0.700
METEOR	**0.563**	**0.559**	**0.425**	**0.706**
chrF++	0.554	0.551	0.423	0.701
$1/2$·METEOR+$1/2$·BLEU NLTK	0.555	0.551	0.421	0.699
$2/3$·METEOR+$1/3$·BLEU NLTK	0.547	0.543	0.419	0.697

Table 7: Ablation study of metrics used as rewards in SCST for t5-large models. The results shown are on the test split.

Learning Rate	BLEU↑	BLEU↑ NLTK	METEOR↑	chrF++↑
10^{-6}	0.553	0.549	0.420	0.698
$5 \cdot 10^{-6}$	**0.558**	**0.554**	**0.422**	**0.700**
10^{-5}	0.544	0.542	0.419	0.696

Table 8: Ablation study on learning rates in SCST (using BLEU NLTK as the optimized metric)

WebNLG G2T Team/model	BLEU↑	BLEU↑ NLTK	METEOR↑	chrF++↑
ReGen G2T.RL.ES.meteor t5-base (early CE)	0.527	0.523	0.413	0.689
ReGen G2T.RL.best.meteor t5-base (best CE)	0.528	0.526	0.412	0.681

Table 9: G2T: Best results for t5-base fine-tuned with SCST using METEOR as reward.

Team Name	BLEU↑	BLEU↑ NLTK	METEOR↑	chrF++↑
ReGen G2T.CE t5-large	0.543±0.007	0.540±0.007	0.416±0.002	0.691±0.002
ReGen G2T.RL t5-large	0.553±0.007	0.550±0.007	0.422±0.002	0.702±0.003
ReGen G2T.CE.ES t5-base (early CE)	0.521±0.004	0.517±0.004	0.404±0.001	0.675±0.002
ReGen G2T.RL.ES t5-base (early CE)	0.528±0.007	0.523±0.007	0.408±0.002	0.682±0.003
ReGen G2T.CE.best t5-base (best CE)	0.524±0.000	0.520±0.001	0.404±0.000	0.670±0.000
ReGen G2T.RL.best t5-base (best CE)	0.525±0.007	0.522±0.007	0.407±0.002	0.681±0.003
ReGen G2T.RL.ES.meteor t5-base (early CE)	0.525±0.007	0.521±0.007	0.412±0.002	0.687±0.003
ReGen G2T.RL.best.meteor t5-base (best CE)	0.527±0.007	0.524±0.007	0.410±0.002	0.686±0.003

Table 10: Results means and standard deviations (SD), shown as mean±SD, for CE and SCST trained models (including our best results model) for a total of 3 different random number generator seeds used in training.

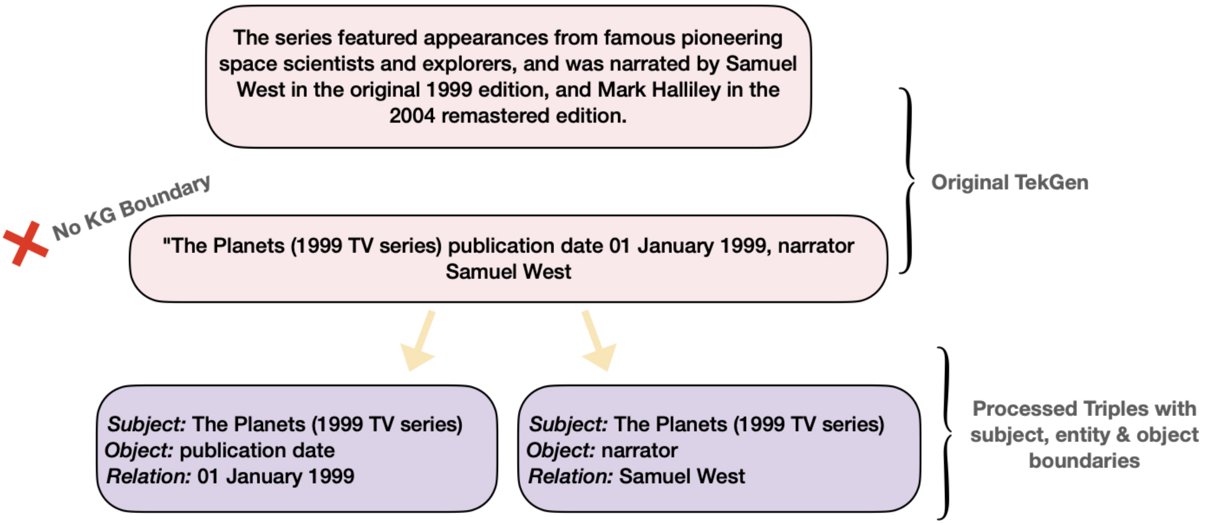
The series featured appearances from famous pioneering space scientists and explorers, and was narrated by Samuel West in the original 1999 edition, and Mark Hallley in the 2004 remastered edition.
Original TekGen
No KG Boundary
"The Planets (1999 TV series) publication date 01 January 1999, narrator Samuel West
Subject: The Planets (1999 TV series)
Object: publication date
Relation: 01 January 1999
Subject: The Planets (1999 TV series)
Object: narrator
Relation: Samuel West
Processed Triples with subject, entity & object boundaries

Figure 3: An example from the processed TEKGEN dataset. The original dataset lacks KG boundaries, which makes it difficult to evaluate T2G systems efficiently.

Type	Sentence / Graph
Source	The Pontiac Rageous began and ended its production in 1997 on an assembly line in Detroit, a city in Michigan.
Gold	Pontiac_Rageous ♢ productionStartYear ♢ 1997 ♢ Pontiac_Rageous ♢ assembly ♢ Michigan ♢ Pontiac_Rageous ♢ assembly ♢ Detroit ♢ Pontiac_Rageous ♢ productionEndYear ♢ 1997 ♢ Detroit ♢ type ♢ City_(Michigan)
Hyp-CE	Pontiac_Rageous ♢ assembly ♢ Detroit ♢ Pontiac_Rageous ♢ modelYears ♢ 1997 ♢ Pontiac_Rageous ♢ modelYears ♢ 1997 ♢ Detroit ♢ isPartOf ♢ Michigan
Hyp-SCST	Pontiac_Rageous ♢ assembly ♢ Detroit ♢ Pontiac_Rageous ♢ modelYears ♢ 1997 ♢ Pontiac_Rageous ♢ assembly ♢ Michigan
Source	In the United States, where Abraham A, Ribicoff was born, African Americans are one of the ethnic groups. Abraham A. Ribicoff was married to Ruth Ribicoff.
Gold	Abraham_A._Ribicoff ♢ spouse ♢ "Ruth Ribicoff" ♢ Abraham_A._Ribicoff ♢ birthPlace ♢ United_States ♢ United_States ♢ ethnicGroup ♢ African_Americans ♢ Abraham_A._Ribicoff ♢ nationality ♢ United_States
Hyp-CE	Abraham_A._Ribicoff ♢ birthPlace ♢ United_States ♢ Abraham_A._Ribicoff ♢ spouse ♢ "Ruth Ribicoff" ♢ United_States ♢ ethnicGroup ♢ African_Americans
Hyp-SCST	Abraham_A._Ribicoff ♢ birthPlace ♢ United_States ♢ Abraham_A._Ribicoff ♢ spouse ♢ "Ruth Ribicoff" ♢ Abraham_A._Ribicoff ♢ nationality ♢ American ♢ United_States ♢ ethnicGroup ♢ African_Americans
Source	Super Capers, edited by Stacy Katzman, is a 98 minute film starring Michael Rooker and Tom Sizemore.
Gold	Super_Capers ♢ editing ♢ Stacy_Katzman ♢ Super_Capers ♢ starring ♢ Michael_Rooker ♢ Super_Capers ♢ starring ♢ Tom_Sizemore ♢ Super_Capers ♢ runtime | 98.0
Hyp-CE	Super_Capers ♢ starring ♢ Tom_Sizemore ♢ Super_Capers ♢ timeOut ♢ "980.0"(minutes) ♢ Super_Capers ♢ starring ♢ Michael_Rooker ♢ Super_Capers ♢ editor ♢ Stacy_Katzman
Hyp-SCST	Super_Capers ♢ starring ♢ Tom_Sizemore ♢ Super_Capers ♢ length ♢ 98.0 (minutes) ♢ Super_Capers ♢ starring ♢ Michael_Rooker ♢ Super_Capers ♢ editor ♢ Stacy_Katzman
Source	Doctor George Cary (1611-1680), Professor of Sacred Theology, lord of the manor of Clovelly, Devon, was Dean of Exeter between 1663 and 1680 (amongst other duties responsible for the maintenance and decoration of Exeter Cathedral).
Gold	George Cary (1611-1680) ♢ position held ♢ Dean of Exeter ♢ start time ♢ 01 January 1663 ♢ date of birth ♢ 00 1611 ♢ date of death ♢ 00 1680
Hyp-CE	George Cary (priest) ♢ date of birth ♢ 01 January 1611 ♢ date of death ♢ 01 January 1680
Hyp-SCST	George Cary (priest) ♢ position held ♢ Dean of Exeter ♢ date of birth ♢ 01 January 1611 ♢ date of death ♢ 01 January 1680
Source	Early general elections were held in the Bahamas on 10 April 1968.
Gold	1968 Bahamian general election ♢ point in time ♢ 10 April 1968 ♢ country ♢ The Bahamas ♢ applies to jurisdiction ♢ The Bahamas
Hyp-CE	1968 Bahamian general election ♢ point in time ♢ 10 April 1968
Hyp-SCST	1968 Bahamian general election ♢ point in time ♢ 10 April 1968 ♢ country ♢ The Bahamas
Source	The school was established on 6 January 1930, by former education minister, CWW Kannangara, who additionally founded two other colleges located in central Ceylon.
Gold	Kattankudy Central College ♢ instance of ♢ School
Hyp-CE	Government Polytechnic , Colombo ♢ inception ♢ 00 1930
Hyp-SCST	Government Polytechnic , Colombo ♢ inception ♢ 00 1930 ♢ instance of ♢ School

Table 11: Few cherry-picked generation for T2G task for WebNLG+ 2020 (top three) and TEKGEN (bottom three). For each source (`Text`), we show the ground truth (`Gold`) and system generated hypothesis from the best CE (`Hyp-CE`) and SCST models (`Hyp-SCST`). Note that the set of triples in WebNLG+ takes the form $x_{\text{G}} = [(s^1 \diamondsuit p^1 \diamondsuit o^1), \ldots, (s^K \diamondsuit p^K \diamondsuit o^K)]$, whereas the same for TEKGEN is of form $x_{\text{G}} = [s \diamondsuit (p^1 \diamondsuit o^1), \ldots, (p^K \diamondsuit o^K)]$

Type	Graph / Sentence
Source	McVeagh_of_the_South_Seas ♢ starring ♢ Harry_Carey_(actor_born_1878) ♢ McVeagh_of_the_South_Seas ♢ writer ♢ Harry_Carey_(actor_born_1878)
Gold	Born in 1878, Harry Carey later grew up to write and star in the movie McVeagh of the South Seas. Harry Carey, born in 1878, wrote and appeared in the movie McVeagh of the South Seas. Harry Carey, who was born in 1878, wrote and starred the film of McVeagh of the South Seas.
Hyp-CE	McVeagh of the South Seas was written by Harry Carey, who was born in 1878.
Hyp-SCST	McVeagh of the South Seas was written by Harry Carey and starred the actor Harry Carey who was born in 1878.
Source	Aleksandr_Prudnikov ♢ height ♢ 185.0 (centimetres) ♢ Aleksandr_Prudnikov ♢ youthclub ♢ FC_Spartak_Moscow ♢ FC_Spartak_Moscow ♢ ground ♢ Otkrytiye_Arena
Gold	Aleksandr Prudnikov, 185cm tall played for FC Spartak Moscow's youth team. FC Spartak Moscow is based in the Otkrytiye Arena. Aleksandr Prudnikov who is 185 cm tall is a member of the youth side of FC Spartak Moscow. The home ground of FC Spartak Moscow is Otkrytiye Arena. Aleksandr Prudnikov is 185.0 cm tall and played for the FC Spartak Moscow at the Otkrytiye Arena.
Hyp-CE	Aleksandr Prudnikov is 185 cm tall and played for FC Spartak Moscow's youth team at the Otkrytiye Arena.
Hyp-SCST	Aleksandr Prudnikov is 185 cm tall and played for the youth team of FC Spartak Moscow whose home ground is the Otkrytiye Arena.
Source	Baku_Turkish_Martyrs'_Memorial ♢ nativeName ♢ "Türk Şehitleri Anıtı" ♢ Baku_Turkish_Martyrs'_Memorial ♢ location ♢ Azerbaijan
Gold	The Native name of the Baku Turkish Martyrs' Memorial is "Türk Şehitleri Anıtı" which is located in Azerbaijan. The native name of the Baku Turkish Martyrs' Memorial is Türk Şehitleri Anıtı located in Azerbaijan. The native name for the Baku Turkish Martyrs' Memorial is Türk Şehitleri Anıtı, which is located in Baku, Azerbaijan.
Hyp-CE	The native name of the Baku Turkish Martyrs' Memorial in Azerbaijan is Türk Şehitleri Anıtı.
Hyp-SCST	The Baku Turkish Martyrs' Memorial is located in Azerbaijan and is known locally as Türk Şehitleri Anıtı.
Source	John Banister (anatomist) ♢ occupation ♢ Surgeon ♢ date of birth ♢ 01 January 1533 ♢ date of death ♢ 01 January 1610
Gold	John Banister (1533-1610) was an English anatomist, surgeon and teacher.
Hyp-CE	John Banister (1533-1610) was an English surgeon.
Hyp-SCST	John Banister (1533-1610) was an English surgeon and anatomist.
Source	WNPT (TV) ♢ country ♢ United States ♢ instance of ♢ Television station
Gold	WNPT, virtual channel 8 (VHF digital channel 7), is a PBS member television station licensed to Nashville, Tennessee, United States.
Hyp-CE	WNPT, virtual channel 3 (UHF digital channel 15), is a Fox-affiliated television station licensed to Portland, Oregon, United States.
Hyp-SCST	WNPT, virtual channel 4 (UHF digital channel 16), is a Public Broadcasting Service (PBS) member television station licensed to Portland, Oregon, United States.
Source	Our Lady of the Presentation Cathedral, Natal ♢ inception ♢ 21 November 1988
Gold	Our Lady of the Presentation Cathedral, Natal was inaugurated on November 21, 1988, and is located in the district of Cidade Alta in Natal, capital of the Brazilian state of Rio Grande do Norte.
Hyp-CE	Our Lady of the Presentation Cathedral, Natal was built in 1988.
Hyp-SCST	Our Lady of the Presentation Cathedral, Natal was consecrated on 21 November 1988.

Table 12: Few cherry-picked generation for G2T task for WebNLG+ 2020 (top three) and TEKGEN (bottom three). For each source (Graph), we show the ground truth (Gold) and system generated hypothesis from the best CE (Hyp-CE) and SCST models (Hyp-SCST). Note that the set of triples in WebNLG+ 2020 takes the form $x_{\text{G}} = [(s^1 \diamondsuit p^1 \diamondsuit o^1), \ldots, (s^K \diamondsuit p^K \diamondsuit o^K)]$, whereas the same for TEKGEN is of form $x_{\text{G}} = [s \diamondsuit (p^1 \diamondsuit o^1), \ldots, (p^K \diamondsuit o^K)]$