Abstract

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

1 Introduction

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

\(^1\)Our code is available at https://github.com/QipengGuo/P2_WebNLG2020.

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

Overall, our contributions are as follows:

- We develop a plan-and-pretrain approach, \( \mathcal{P}^2 \), for knowledge graph-to-text generation. \( \mathcal{P}^2 \) unleashes the power of the pretrained T5 model by pipelining it with a R-GCN content planner and special canonicalization rules.

- Our \( \mathcal{P}^2 \) model achieved the top-1 performance on all criteria of both the automatic and human evaluations of knowledge graph-to-text generation task in English.
• We open-sourced our code and outputs as a reference for future work.

2 WebNLG 2020 Challenge Overview

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

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

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

<table>
<thead>
<tr>
<th>Split</th>
<th>Number of Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train (T-G Pairs)</td>
<td>35,426</td>
</tr>
<tr>
<td>Valid (T-G Pairs)</td>
<td>4,464</td>
</tr>
<tr>
<td>Test (Only T, for T2G)</td>
<td>2,155</td>
</tr>
<tr>
<td>Test (Only G, for G2T)</td>
<td>1,779</td>
</tr>
</tbody>
</table>

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

3 \(P^2\) for G2T Generation

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

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

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

3.1 R-GCN Planner

Our R-GCN planner is largely based on Zhao et al. (2020). The input of the planner is a knowledge graph as represented via the Resource Description Framework (RDF) (Lassila et al., 1998), where the basic unit is the triplet of a subject (s), predicate (p), and object (o), such as (Paris, IsCapitalOf, France). Since there can be multiple triplets (e.g., 2-7 triplets as in the WebNLG dataset), the planner takes in a random order of triplets 1 to N, and aims to find an ideal order of the triplets, such as “(s2, p2, o2) \(\rightarrow\) (s3, p3, o3) \(\rightarrow\) (s1, p1, o1).” The

2https://webnlg-challenge.loria.fr/challenge_2020/
ideal plan we want to generate corresponds to the order that the triplets are mentioned in the ground-truth text.

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

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

3.2 Converting Plans to Sequences

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

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

3.3 Pretrained Seq2Seq Model for Plan-to-Text Generation

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

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

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

3.4 Canonicalization of Special Tokens

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

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

---

3https://github.com/ThiagoCF05/webnlg/tree/master/data/v1.5
Table 2: Performance of top 10 systems and two official baselines (Base1 and Base 2) on the leaderboard of the WebNLG 2020 English RDF-to-text challenge as ranked by METEOR. With all metrics, larger is better, with the exception of TER where lower is better. In all categories, our system achieved the first place. Note that ID28 is an unsupervised system (Guo et al., 2020).

<table>
<thead>
<tr>
<th>ID</th>
<th>BLEU</th>
<th>METEOR</th>
<th>chrF++</th>
<th>TER</th>
<th>BERT_Precision</th>
<th>BERT_Recall</th>
<th>BERT_F1</th>
<th>BLUERT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID18 (Ours)</td>
<td>53.98</td>
<td>0.417</td>
<td>0.690</td>
<td>0.406</td>
<td>0.960</td>
<td>0.957</td>
<td>0.958</td>
<td>0.62</td>
</tr>
<tr>
<td>ID30</td>
<td>53.54</td>
<td>0.414</td>
<td>0.688</td>
<td>0.416</td>
<td>0.958</td>
<td>0.955</td>
<td>0.956</td>
<td>0.61</td>
</tr>
<tr>
<td>ID30_1</td>
<td>52.07</td>
<td>0.413</td>
<td>0.685</td>
<td>0.444</td>
<td>0.955</td>
<td>0.954</td>
<td>0.954</td>
<td>0.58</td>
</tr>
<tr>
<td>ID34</td>
<td>52.67</td>
<td>0.413</td>
<td>0.686</td>
<td>0.423</td>
<td>0.957</td>
<td>0.955</td>
<td>0.956</td>
<td>0.6</td>
</tr>
<tr>
<td>ID5</td>
<td>51.74</td>
<td>0.411</td>
<td>0.679</td>
<td>0.435</td>
<td>0.955</td>
<td>0.954</td>
<td>0.954</td>
<td>0.6</td>
</tr>
<tr>
<td>ID35</td>
<td>51.59</td>
<td>0.409</td>
<td>0.681</td>
<td>0.431</td>
<td>0.956</td>
<td>0.954</td>
<td>0.954</td>
<td>0.59</td>
</tr>
<tr>
<td>ID23</td>
<td>51.74</td>
<td>0.403</td>
<td>0.669</td>
<td>0.417</td>
<td>0.959</td>
<td>0.954</td>
<td>0.956</td>
<td>0.61</td>
</tr>
<tr>
<td>ID2</td>
<td>50.34</td>
<td>0.398</td>
<td>0.666</td>
<td>0.435</td>
<td>0.954</td>
<td>0.950</td>
<td>0.951</td>
<td>0.57</td>
</tr>
<tr>
<td>ID15</td>
<td>40.73</td>
<td>0.393</td>
<td>0.646</td>
<td>0.511</td>
<td>0.940</td>
<td>0.946</td>
<td>0.943</td>
<td>0.45</td>
</tr>
<tr>
<td>ID28</td>
<td>44.56</td>
<td>0.387</td>
<td>0.637</td>
<td>0.479</td>
<td>0.949</td>
<td>0.949</td>
<td>0.948</td>
<td>0.54</td>
</tr>
<tr>
<td>Base1</td>
<td>40.22</td>
<td>0.384</td>
<td>0.648</td>
<td>0.476</td>
<td>0.949</td>
<td>0.950</td>
<td>0.949</td>
<td>0.55</td>
</tr>
<tr>
<td>Base2</td>
<td>38.07</td>
<td>0.367</td>
<td>0.626</td>
<td>0.515</td>
<td>0.933</td>
<td>0.941</td>
<td>0.932</td>
<td>0.50</td>
</tr>
</tbody>
</table>

4 Experiments

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

4.1 Implementation Details

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

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

For the knowledge graph-to-text generation task, WebNLG 2020 evaluates the text quality by a range of different quantitative metrics (Moussalem et al., 2020) that are listed on an automatic evaluation leaderboard. For this purpose, the quality of textual outputs are assessed using BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), chrF++ (Popovic, 2017), and TER (Snover et al., 2006). In brief, these metrics quantify the n-gram recall, precision, or F-scores between the model outputs and the ground-truth references. Additionally, several BERT-based scores are also reported, including BERT_Precision, BERT_Recall, and BERT_F1 from Zhang et al. (2020), and BLUERT from Sellam et al. (2020).

We present the performance of the top 10 systems and two official baselines on the leaderboard of WebNLG 2020 English RDF-to-text challenge in Table 2. Our P² model achieves the highest out of all systems on all automatic evaluation criteria, indicating higher similarity to the ground truth human written reference in the test set. For example, in terms of BLEU our model outperforms the second system by +0.44, and is over the two official baselines by +13.76 and +15.91, respectively.
### 4.3 Human Evaluation

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

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

### 5 A Simple Baseline for T2G Extraction

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


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


