Enriching Data-to-Text Generation Models with Hierarchical Ontologies

Ashish Bharadwaj Srinivasa
ashishbs7@gmail.com

Abstract

Most approaches for data-to-text conversion work with flat ontological structures by linearizing the table cells and using seq2seq language models for text generation. In this work, we demonstrate that given a set of highlighted table cells, incorporating rich hierarchical ontologies present in tables as semantic triples generates more semantically sound text. We fine-tune state-of-the-art pretrained language models for data-to-text generation by enriching the input text with hierarchical semantic inter-dependencies. We compare the performance of various linearization strategies and pretrained language model combinations and study the impact both on BLEU, METEOR and semantic matching learned metrics such as BLEURT and BERTScore. We show that our approach of linearizing hierarchical ontologies improves the data-to-text generation on the DART corpus.

1 Introduction

Data-to-Text generation has gained a lot of interest in recent years due to its direct applicability to question answering systems, chatbots, summarization, etc. Moreover, most enterprise data is highly structured and stored in the form of databases and knowledge bases. The natural language task involves converting structured data, typically in tables or knowledge bases, into natural language text. Data-to-Text has become a critical driver of the AI revolution of the recent years as it actively helps address NLP problems such as question answering, chat bots (dialog), machine translation, etc. As huge amounts of data in businesses is generally stored in large structured databases, extending natural language generation techniques to them requires carefully fine tuned algorithms. Moreover, the databases can be from a wide variety of domains and thus finding generic techniques that work well becomes a challenging problem. There are two major ways to approach the problem - pipeline and end-to-end based approaches. In the pipeline based models, the table-to-text generation task is broken down into multiple linked smaller tasks such as content selection, lexicalization, text ordering, etc. Transfer learning has enabled the usage of pretrained language models trained on massive amounts of text data to be used for data-to-text generation. With this, end-to-end text generation approaches have become feasible where the need for learning each of the pipeline steps is reduced and left to be learned by the deep neural network models itself. Two of the more recent end-to-end generation techniques involve using raw highlighted table cells or using semantic triples extracted from tables as the input for natural language generation algorithms. In 2020 there have been multiple popular datasets published for these two approaches - DART (Radev et al., 2020) contain triples to text examples and ToTTo (Parikh et al., 2020) contains examples of highlighted table cells to text. Most state of the art data-to-text approaches completely ignore the hierarchical inter-dependencies in the tables and linearize the semantic triples into flat representation. These approaches fail to consider the hierarchical nature of the data and the dependencies.

The central hypothesis of this paper is that using linearization strategies specialized for handling the hierarchical semantic inter-dependencies leads to more semantically and syntactically accurate text. Secondly, most data-to-text approaches use an ordered linearization scheme which limits the learnings from the model. In this work, we show that permutation invariant linearization strategies are highly effective for text generation. For our work, we use pretrained language models such as BART, Bert2Bert, T5, etc. and customize or fine tune them on the DART dataset. Finally, we evaluate the models using BLEU, METEOR, BLEURT

and BERTScore metrics.

Our contribution to data-to-text generation is the following:

- Demonstrating the importance of pretrained language models in open domain end-to-end data-to-text generation.
- Exploring linearization strategies to incorporate hierarchical ontological inter-dependencies inherent in the data.
- Showing the value of permutation invariant linearization strategies for data-to-text model fine-tuning.

2 Related Work

There have been a number of approaches to solving data-to-text generation. (Gehrmann et al., 2018) used the idea of pointer generator networks (See et al., 2017) for data-to-text generation from meaning representations. (Bao et al., 2018) extended the above idea of pointer generator networks and introduced an additional copying mechanism. This proved highly effective for data-to-text generation tasks due to this task's inherent likelihood of having the same input text in the generated text. (Ferreira et al., 2019) compared pipeline and end-to-end based approaches for data-to-text generation and also showed the value of using transformer (Vaswani et al., 2017) based models for the same. There have also been approaches using CTRL (Keskar et al., 2019), which is specifically trained for conditional text generation.

With the advent of pretrained language models trained on massive amounts of data, the most recent approaches have utilized them for text generation. These models perform really well at open domain text generation, but require fine tuning to handle the linearized data or table inputs. Many popular architectures have proven to be effective for data-to-text generation such as BART (Lewis et al., 2019), Bert2Bert (Rothe et al., 2020), T5 (Kale, 2020). Most of these approaches have suffered from generating semantically sound sentences when used to generate open domain text. To tackle this issue, (Harkous et al., 2020) introduced the idea of a semantic fidelity classifier which penalizes the generated text if it is not semantically sound. There have also been many learned metrics such as BERTScore (Zhang et al., 2020) and BLEURT (Sellam et al., 2020) which use pretrained language models to make semantic similarity judgements. We use these metrics to evaluate our models.

3 Data

(Radev et al., 2020) curated DART - an open-domain data-to-text dataset gathered from highlighted table cells. The data is in the form of semantic RDF triples and their corresponding verbalized representation. What makes DART unique and conducive to prove our hypotheses is its emphasis on capturing hierarchical ontological structures. While previous table-to-text corpora such as ToTTo (Parikh et al., 2020), WikiTableText (Bao et al., 2018), etc. capture only the flat ontological structures of table rows and columns headers, DART captures rich semantic inter-dependencies among these table cells as RDF triples. Below, we can see an example hierarchical dependency among table cells shown in DART.

```
Abilene Regional Airport
--> CITY_SERVED:Abilene, Texas
     --> IS_PART_OF:Taylor County, Texas
     --> COUNTRY:United States
```

Additionally, DART also places strong emphasis on its open-domain nature. While popular datasets like WebNLG, E2E contain data from limited number of semantic domains, DART captures tables extracted from Wikipedia tables in general. This poses an added challenge and forces the models to be generalized enough to handle a wide variety of domains. The data is pre-split into train, dev and test sets with each data point containing a list of semantic triples in JSON/XML format carefully collected by a combination of human and automatic annotation. There is a total of 82,191 examples generated from a total of 5,623 tables. The two main challenges posed by DART as described above are due its open domain nature and hierarchical ontology structure. This leads to popular state-of-the-art table-to-text methods being less effective as they only consider a linearized (or flattened) semantic representation. Below is an example of a data sample in the DART dataset in the JSON structure.


```
{
  "tripleset": [
    [
      "Ben Mauk",
      "High school",
      "Kenton"
    ],
    [
      "Ben Mauk",
      "College",
```

split	depth = 1	depth >1	total
train	50182 (80%)	12477 (20%)	62659
dev	5895 (85%)	1085 (15%)	6980
test	9432 (75%)	3120 (25%)	12552

Table 1: Comparison of input ontology depths in different splits of DART.


```
      "Wake Forest Cincinnati"
    ]
  ],
  "subtree_was_extended": false,
  "annotations": [
    {
      "source":
          "WikiTableQuestions_lily",
      "text": "Ben Mauk, who attended
          Kenton High School, attended
          Wake Forest Cincinnati for
          college."
    }
  ]
}
```


We also look at the depth of the ontologies in the DART dataset in Table 1 as we want to demonstrate the impact of various models on the different slices. As we can see, the number of triple sets with depth greater than 1 is higher in the dev and test sets compared to the training set. This poses an additional unseen challenge for the models on the dev and test sets when compared to the training set. Thus, due to the highly open domain and the hierarchical nature of the DART dataset, it is best suited for our work to build a generalized data-to-text model enriched with semantic interdependencies in the table.

4 Methodology

4.1 Models

We adopt the end-to-end data-to-text generation strategy for this work where we convert an input set of triples into text without any intermediate pipeline planning steps. We use pre-trained transformer based language models due to their superior text generation capability. These models have been trained on large corpora of web data and thus it gives us a good platform to build on top of instead of training from scratch. We leverage transfer learning to fine tune these pre-trained language models on the DART dataset and the data-to-text generation task. The intuition for this is that the pre-trained language models have learned good feature engineering layers which can be leveraged as is. In our experiments, we do not tweak the layers we fine tune from these models, instead fine tuning all of them. Specifically, we use the following models:

- BART (Lewis et al., 2019) → A denoising autoencoder model pretrained for sequence-to-sequence modeling. This is the model that performed the best on DART in the original work (Radev et al., 2020) and we use this as the baseline model for our analysis in this work. For the experiments, we use BART-base variant which has 12 layers and a total of 139M parameters.

- T5 (Kale, 2020) → T5 is a text-to-text model that has shown state-of-the-art performance when fine tuned for data-to-text generation on other datasets. We adapt and fine tune this model for DART. We use the T5-small variant due to compute limitations which has 6 layers and a total of 60M parameters.

- Bert2Bert (Rothe et al., 2020) → And encoder decoder model with the encoder and decoder initialized with BERT-base weights. We add additional cross attention layers from the decoder to the encoder and learn these weights while we finetune the pretrained weights for the task. As the cross attention weights are randomly initialized, this model will require a higher number of gradient update steps to converge. This model is also considerably bigger as we combine two BERT models together.

- Roberta2Roberta (Rothe et al., 2020) → Similar to Bert2Bert, with the encoder and decoder replaced with Roberta.

4.2 Linearization

In order to use any kind of language modeling approach for data-to-text generation, including the models described above, we need to first convert the input ontology or table cells into a linear text form. Most table-to-text approaches use a flattened representation of the ontology as described in (Ferreira et al., 2019). We adapt the proposed

linearization strategy to convert triple sets into text for DART. The advantage of this approach is that it provides an easy way to convert ontologies and tables into text, which can then be fed into any state-of-the-art seq2seq language models. The limitation of this approach is that there is no information in the linearized representation regarding the semantic inter-dependencies of the hierarchical ontology present in the DART corpus.

```
[TABLECONTEXT]
--> [TITLE]:List of canals of the
    United Kingdom
--> CANAL:Stevenston Canal
     --> YEAR_OPENED:1772
     --> LOCKS:0
     --> LENGTH_(MILES):2.25
```

Listing 1: Example ontology input in DART

```
The 2.25-mile long Stevenston Canal in
    the UK opened in 1772 and had no
    locks.
```

Listing 2: Reference translation for the above ontology

Given an input ontology as shown in Listing 1 we describe the three linearization strategies we explore and introduce in our experiments to generate text. The reference text for the example is shown in Listing 2 and we can observe that to rightly generate the text, the model needs to correctly parse the nested relationship structure of the input ontology.

Strategy 0

In the first strategy adapted from previous approaches we flatten the ontology into a list of triples. We preserve the input ordering of the triples in the dataset. Each triple is linearized with the start and end tags identifying a triple unit and ordered as subject, predicate, object. The linearization obtained for the above ontology is shown below.

```
<TRIPLE> [TABLECONTEXT] CANAL
    Stevenston Canal </TRIPLE> <TRIPLE>
    Stevenston Canal YEAR_OPENED 1772
    </TRIPLE> <TRIPLE> [TABLECONTEXT]
    [TITLE] List of canals of the
    United Kingdom </TRIPLE> <TRIPLE>
    Stevenston Canal LOCKS 0 </TRIPLE>
    <TRIPLE> Stevenston Canal
    LENGTH_(MILES) 2.25 </TRIPLE>
```

Strategy 1

One of the limitation of the naive flattened strategy above is that it biases towards input ordering of the triples. Ideally, we want to learn a permutation invariant triple set to text generation model and to do this we augment our training dataset with permutations of the triples. Each such list of triples is linearized similar to Strategy 0 and fed into the pre-trained language model. Each of the linearizations are trained with the same reference text label. For performance reasons, we limit the number of such permutations to 3. The linearizations obtained from this strategy are shown below.

```
<TRIPLE> [TABLECONTEXT] CANAL
    Stevenston Canal </TRIPLE> <TRIPLE>
    Stevenston Canal YEAR_OPENED 1772
    </TRIPLE> <TRIPLE> [TABLECONTEXT]
    [TITLE] List of canals of the
    United Kingdom </TRIPLE> <TRIPLE>
    Stevenston Canal LOCKS 0 </TRIPLE>
    <TRIPLE> Stevenston Canal
    LENGTH_(MILES) 2.25 </TRIPLE>

<TRIPLE> Stevenston Canal YEAR_OPENED
    1772 </TRIPLE> <TRIPLE> Stevenston
    Canal LOCKS 0 </TRIPLE> <TRIPLE>
    Stevenston Canal LENGTH_(MILES)
    2.25 </TRIPLE> <TRIPLE>
    [TABLECONTEXT] CANAL Stevenston
    Canal </TRIPLE> <TRIPLE>
    [TABLECONTEXT] [TITLE] List of
    canals of the United Kingdom
    </TRIPLE>

<TRIPLE> Stevenston Canal LOCKS 0
    </TRIPLE> <TRIPLE> Stevenston Canal
    LENGTH_(MILES) 2.25 </TRIPLE>
    <TRIPLE> [TABLECONTEXT] CANAL
    Stevenston Canal </TRIPLE> <TRIPLE>
    Stevenston Canal YEAR_OPENED 1772
    </TRIPLE> <TRIPLE> [TABLECONTEXT]
    [TITLE] List of canals of the
    United Kingdom </TRIPLE>
```

Strategy 2

Though Strategy 1 tackles the permutation invariant nature of data-to-text generation, it still flattens the ontology structure in its linearization. We propose a new linearization scheme for hierarchical ontologies by converting tripleset into an ontology tree. We define a context free grammar to convert the ontology tree into a nested linearized text.

```
linearized_input -> children

children -> child*

child -> <CHILD> Predicate Object
    <CHILDREN> children </CHILDREN>
    </CHILD>
```

Using the linearization strategy, we hypothesize that the model will better learn the semantic inter-dependencies and thereby generate more seman-

tically accurate text. This strategy will be more effective on more complex ontological inputs in the DART corpus. Below is the linearization obtained from this strategy for the example ontology described in Listing 1

```
<CHILD> HEAD [TABLECONTEXT] <CHILDREN>
    <CHILD> CANAL Stevenston Canal
    <CHILDREN> <CHILD> YEAR_OPENED 1772
    <CHILDREN> </CHILDREN> </CHILD>
    <CHILD> LOCKS 0 <CHILDREN>
    </CHILDREN> </CHILD> <CHILD>
    LENGTH_(MILES) 2.25 <CHILDREN>
    </CHILDREN> </CHILD> </CHILDREN>
    </CHILD> <CHILD> [TITLE] List of
    canals of the United Kingdom
    <CHILDREN> </CHILDREN> </CHILD>
    </CHILDREN> </CHILD>
```

5 Experiments

5.1 Setup

In our experiments, we use the DART dataset and explore the impact of the different pre-trained models coupled with the various linearization strategies described above. Due to compute limitations, we fine tune each of these models on a maximum of 5 epochs and perform limited hyperameter search to optimize for the learning rate and the schedules. The DART dataset is parsed and linearizations are extracted for each strategy without modifying the reference text translation. We use the best performing model on the DART corpus as our baseline for the experiments. Firstly, we retrain the baseline model to find the metrics given our compute, learning rate, batch size, etc. Once the baseline model was fine tuned and metrics established, we fine tune a new state-of-the-art text-to-text generation model T5 for the DART data-to-text generation task with the same linearization strategy. Next we explore encoder decoder models built using pretrained language modeling checkpoints from BERT and RoBERTa. These models have randomly initialized cross attention layers from the decoder to the encoder. Thus they require more training steps to perform well. Due to limited compute resources, we train all the models for the same number of epochs. Additionally, we optimize the models using automatic mixed precision for training on GPU with CUDA. We fine tune BART-base and T5-small for the newly proposed linearization strategies Strategy 1 and Strategy 2 and compare and study the metrics.

5.2 Metrics

For the data-to-text generation task described above, we use two types of metrics - sequence correlation based metrics and semantic match/distance metrics. Improving both of these two types of metrics is critical for the task we are solving and the hypotheses proposed. The quality of any text generation task will first need to be evaluated on a purely token matching basis to measure how correlated the generated text sequence is to the true/expected sequence. These can be captured by using the popular sequence matching metrics such as BLEU (Papineni et al., 2002) and METEOR (Banerjee and Lavie, 2005). Both the metrics are a modified version of the precision metric that allows comparing the predicted text with the reference text on a token co-occurrence level. While, BLEU is a popularly used metric that most state-of-the-art models report on, we additionally use METEOR as it tackles some of the problems with BLEU. Specifically, METEOR considers matching on word stems as opposed to exact token match and also allows for comparing sentences directly as opposed to corpora.

Secondly, we measure how semantically close the generated text is to the reference text. This is even more important for our specific hypotheses as we want to measure if the rich semantic representation from the hierarchical ontologies translated to the generated text. Pure token matching approaches might miss the semantic distinction from hierarchical information, due to their inherent unordered nature. To track this we use BERTScore (Zhang et al., 2020) and BLEURT (Sellam et al., 2020). These are machine learned metrics which compare a candidate sentence to a reference sentence by computing similarity at a token level using contextual embeddings as opposed to exact matches like the previous two metrics. This allows the metrics to capture rich semantic similarity assessments and makes them very effective for evaluating our hypothesis. Additionally, as these metrics are fully differentiable there is scope to use these metrics as part of the loss function to optimize the model as well. We leave this as future work.

6 Results and Analysis

The BART model we fine tune performs slightly better on the test set compared to the baseline model presented in (Radev et al., 2020). We use this as the baseline for our comparisons. Since

Model	Linearization Strategy	DART - dev BLEU	METEOR	BLEURT mean	BERTScore F1	DART - test BLEU	METEOR	BLEURT mean	BERTScore F1
BART (original paper)	-	-	-	-	-	-	0.36	0.22	0.92
BART	0	36.32	0.58	0.46	0.94	30.57	0.52	0.37	0.93
T5	0	32.41	0.54	0.40	0.93	27.49	0.48	0.30	0.93
Bert2Bert	0	15.64	0.34	0.09	0.85	14.54	0.32	0.08	0.84
Roberta2Roberta	0	13.24	0.32	0.08	0.84	13.11	0.30	0.08	0.82
BART	1	35.66	0.58	0.43	0.94	30.04	0.51	0.35	0.93
T5	1	34.33	0.56	0.43	0.94	29.21	0.50	0.34	0.93
BART	2	36.45	0.58	0.46	0.94	30.93	0.52	0.37	0.93
T5	2	32.16	0.54	0.40	0.93	27.31	0.48	0.30	0.92

Table 2: Model results on the DART dev and test sets comparing various modeling strategies. Higher is better for all the metrics listed.

Model	Linearization Strategy	depth = 1 BLEU	METEOR	BLEURT mean	BERTScore F1	depth >1 BLEU	METEOR	BLEURT mean	BERTScore F1
BART	0	31.57	0.54	0.40	0.94	28.04	0.47	0.30	0.92
T5	0	28.51	0.50	0.34	0.93	24.91	0.43	0.20	0.91
BART	1	31.06	0.53	0.37	0.93	27.46	0.46	0.27	0.92
T5	1	30.08	0.52	0.37	0.93	27.01	0.45	0.26	0.92
BART	2	31.99	0.55	0.40	0.94	28.57	0.49	0.33	0.94
T5	2	28.44	0.50	0.33	0.93	25.56	0.44	0.24	0.93

Table 3: Model results on the DART test sets comparing various modeling strategies sliced by the depth of the input tree. Higher is better for all the metrics listed.

BART is a state-of-the-art model for text generation, we compare the results with T5, which is a state-of-the-art model for text-to-text generation fine tuned for data-to-text generation. In our experiments, BART performs slightly better than T5 with the caveat that we only used T5-small due to compute constraints. The performance is predicted to be up by a few percentage points if we use T5 models with more layers such as T5-Large and T5-3B. Additionally, we also observe that training T5 for longer number of steps produces better results than when compared to a fixed 5 epochs. Secondly, we observe that encoder-decoder models from pre-trained BERT and RoBERTa models do not perform as well as BART and T5 in our setup. As these have a huge number of randomly initialized cross attention weights, they require more training data and number of epochs to produce better text generation. The different model and linearization variants and their associated metrics are listed in Table 2.

We find that Strategy 1 performs slightly better than Strategy 1 for the dev and test sets for T5 and BART models although this model takes longer to train due to the augmented training data from the permutations. From qualitative analysis we also find that the models trained with Strategy 1 produce permutation invariant data-to-text generation. We can extend this qualitative analysis to a quantitative metric in the future.

The hierarchical linearization strategy proposed in Strategy 2 proves effective on generating text for ontologies with a tree depth greater than 1. This can be observed in Table 3 where we compare the performance of the models on flat ontologies with deeper ontologies. T5 and BART models produce more semantically accurate texts when trained with Strategy 2 as can be evidenced by the higher BLEURT and BERTScore numbers when compared with Strategy 1 and Strategy 0.

Finally, we do manual analysis to assess the quality of the text generated by the various approaches. The text generated by BART and T5 looks very semantically sound and fluent as they are pretrained on huge corpora of text. Whereas the text generated from Bert2Bert and Roberta2Roberta contain repeated tokens and are longer in length. We also observe that training the encoder-decoder models for longer steps produces more fluent text. We observe that text generated from Strategy 2 captures

the nested ontology structure better for ontologies with higher depth. This proves that our hierarchical linearization strategy is effective for capturing the semantic inter-dependencies. We also notice that the generated text sometimes contains hallucination which is the occurrence of text that is not present in the input data. This is a widely observed problem in using pretrained language models for data-to-text generation. We hope to address this in the future by defining a penalty parameter for adding new tokens in the generated text.

7 Conclusion

In this work, we showed the applicability of fine tuning pretrained language models for open domain data-to-text generation. We analyzed the limitations of existing approaches for handling hierarchical semantic dependencies existing in the tables. To solve this we propose a new linearization strategy that captures the hierarchical nature of the triples in text form. We see that our fine tuned BART and T5 models perform better for ontologies with a deeper tree structure. We also showed that effective permutation invariant training of data-to-text models generalizes the model to order independence. In the future, we hope to make architectural changes to the pretrained language models to handle hierarchical ontologies removing the need to flatten or linearize the text. Additionally, we also plan to define a learned classifier for assigning a penalty when the generated text fails to capture the dependencies in the input data.

Acknowledgments

We would like to thank (Radev et al., 2020) for curating an exceptional data-to-text generation dataset incorporating hierarchical ontologies. This dataset helped provide a direction and motivation for this work. We would also like to thank (Wolf et al., 2020) for helping the NLP and machine learning communities by simplifying the task of transfer learning on pretrained language models. Finally, we would like to thank Professor Christopher Potts and Bill MacCartney and the teaching staff of Stanford's CS224U for providing project guidance and driving Natural Language Understanding best practices.

Author Contributions

Ashish Bharadwaj Srinivasa

- Reviewed techniques and best practices used for data-to-text generation and identified pitfalls of current approaches.
- Explored and selected the dataset for the work and performed exploratory data analysis on the same.
- Tested baselines explored in the original work and established parity.
- Developed new linearization strategies for hierarchical ontologies.
- Developed new techniques to make data-to-text permutation invariant.
- Customized and fine-tuned pretrained language models for the data-to-text generation task.
- Evaluated the models and performed a comparative quantitative analysis on metrics such as BLEU, METEOR, BLEURT and BERTScore.
- Documented findings in this paper for future research.

References

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

Junwei Bao, Duyu Tang, Nan Duan, Zhao Yan, Yuanhua Lv, Ming Zhou, and Tiejun Zhao. 2018. Table-to-text: Describing table region with natural language. *CoRR*, abs/1805.11234.

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.

Sebastian Gehrmann, Falcon Dai, Henry Elder, and Alexander Rush. 2018. End-to-end content and plan selection for data-to-text generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 46–56, Tilburg University, The Netherlands. Association for Computational Linguistics.

Hamza Harkous, Isabel Groves, and Amir Saffari. 2020. Have your text and use it too! end-to-end neural data-to-text generation with semantic fidelity.

Mihir Kale. 2020. Text-to-text pre-training for data-to-text tasks.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation.

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.

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, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Ankur P Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. ToTTo: A controlled table-to-text generation dataset. In *Proceedings of EMNLP*.

Dragomir Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Nazneen Fatema Rajani, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, Yangxiaokang Liu, Nadia Irwanto, Jessica Pan, Faiaz Rahman, Ahmad Zaidi, Murori Mutuma, Yasin Tarabar, Ankit Gupta, Tao Yu, Yi Chern Tan, Xi Victoria Lin, Caiming Xiong, and Richard Socher. 2020. Dart: Open-domain structured data record to text generation. *arXiv preprint arXiv:2007.02871*.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging pre-trained checkpoints for sequence generation tasks.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert.