# How Do Seq2Seq Models Perform on End-to-End Data-to-Text Generation?

**Xunjian Yin** and **Xiaojun Wan**
Wangxuan Institute of Computer Technology, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
{xjyin,wanxiaojun}@pku.edu.cn

## Abstract

With the rapid development of deep learning, Seq2Seq paradigm has become prevalent for end-to-end data-to-text generation, and the BLEU scores have been increasing in recent years. However, it is widely recognized that there is still a gap between the quality of the texts generated by models and the texts written by human. In order to better understand the ability of Seq2Seq models, evaluate their performance and analyze the results, we choose to use Multidimensional Quality Metric(MQM) to evaluate several representative Seq2Seq models on end-to-end data-to-text generation. We annotate the outputs of five models on four datasets with eight error types and find that 1) copy mechanism is helpful for the improvement in Omission and Inaccuracy Extrinsic errors but it increases other types of errors such as Addition; 2) pre-training techniques are highly effective, and pre-training strategy and model size are very significant; 3) the structure of the dataset also influences the model's performance greatly; 4) some specific types of errors are generally challenging for seq2seq models.

## 1 Introduction

Data-to-text generation is a task of automatically producing text from non-linguistic input (Gatt and Krahmer, 2018). The input can be in various forms such as databases of records, spreadsheets, knowledge bases, simulations of physical systems.

Traditional methods for data-to-text generation (Kukich, 1983; Reiter and Dale, 2000; Mei et al., 2015) implement a pipeline of modules including content planning, sentence planning and surface realization. Recent neural generation systems (Lebret et al., 2016; Wiseman et al., 2017a) are trained in an end-to-end fashion using the very successful encoder-decoder architecture (Bahdanau et al., 2014) as their backbone. Ferreira et al. (2019) introduce a systematic and comprehensive comparison between pipeline and end-to-end architectures for this task and conclude that the pipeline models can generate better texts and generalize better to unseen inputs than end-to-end models.

However, with the rapid development of the Seq2Seq models especially pre-trained models, more and more end-to-end architectures based on Seq2Seq paradigm get state-of-the-art results on data-to-text benchmarks nowadays. Although BLEU score (Papineni et al., 2002), which is based on precision, has been improved dramatically on standard data-to-text benchmarks such as WebNLG (Gardent et al., 2017), ToTTo (Parikh et al., 2020) and RotoWire (Wiseman et al., 2017b) over the recent years, it is commonly accepted that, compared with human evaluation, BLEU score can not evaluate the models very well. It is too coarse-grained to reflect the different dimensions of the models' performance and not always consistent with human judgment (Novikova et al., 2017a; Reiter, 2018; Sulem et al., 2018). Moreover, existing human evaluations on data-to-text generation are usually limited in size of samples, numbers of datasets and models, or dimensions of evaluation.

In this study, we aim to conduct a thorough and reliable manual evaluation on Seq2Seq-based end-to-end data-to-text generation based on multiple datasets and evaluation dimensions. We want to know the pros and cons of different Seq2Seq models on this task, and the factors influencing the generation performance. Particularly, following Multidimensional Quality Metric(MQM) (Mariana, 2014), similar to the job on summarization evaluation (Huang et al., 2020), we use 8 metrics on the Accuracy and Fluency aspects to count errors, respectively. Therefore, compared with existing manual evaluation reports, it is more informative and objective.

Using this method, we manually evaluate several representative models, including Transformer (Vaswani et al., 2017), Transformer with Pointer



*Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*
*Volume 1: Long Papers*, pages 7701 - 7710

Generator (See et al., 2017), T5(small&base) (Raffel et al., 2019), BART(base) (Lewis et al., 2019) [1]. We test these models on four common datasets, including E2E (Novikova et al., 2017b), WebNLG (Gardent et al., 2017), WikiBio (Lebret et al., 2016), ToTTo (Parikh et al., 2020). Thus we can discuss the effectiveness of the pre-training method, some essential techniques and the number of parameters. We can also compare the differences between datasets and how they influence the models' performance. Empirically, we find that:

1. Pre-training: Pre-training is powerful and effective which highly increases the ability of the Seq2Seq paradigm on the data-to-text task.

2. Size: The size of the model makes difference to the results. Particularly, T5-base achieves the best scores on both automatic and human evaluations.

3. Essential Techniques: The copy mechanism can make noticeable improvements for the basic Seq2Seq model, decreasing word-level errors such as Omission and Inaccuracy Extrinsic. But it also introduces more Addition errors slightly.

4. Dataset Structure: The structure of the dataset also influences the model's understanding of the sequence greatly. Content-controlled generation is still a little hard for the Seq2Seq models.

5. Error Type: The most common mistakes of Seq2Seq models on data-to-text task are Omission, Inaccuracy Intrinsic and Inaccuracy Extrinsic, indicating the direction we need to improve the effectiveness of the model. On the other hand, models perform well in fluency.

## 2 Related Work

**Data-to-Text Generation** Traditional methods for data-to-text generation (Kukich, 1983; Mei et al., 2015) implement a pipeline of modules including content planning, sentence planning and surface realization. Recent neural generation systems (Lebret et al., 2016; Wiseman et al., 2017a) are trained in an end-to-end fashion using the very successful encoder-decoder architecture (Bahdanau et al., 2014) as their backbone. Many Seq2Seq models have demonstrated their effectiveness on data-to-text tasks. Since we want to make a general comparison on Seq2Seq models, we will focus on this method. Moreover, with the development of pre-training methods, more and more work (Kale, 2020; Wang et al., 2021; Kale and Rastogi, 2020) began to introduce pre-training model for data-to-text generation.

There is some work evaluating and analyzing the data-to-text generation task. Perez-Beltrachini and Gardent (2017) propose a methodology to analyze the data-to-text benchmarks and apply their method to WikiBio, RNNLG (Wen et al., 2016) and IMAGEDESC (Novikova and Rieser, 2016) datasets. Ferreira et al. (2019) introduce a systematic comparison between pipeline and end-to-end architectures for neural data-to-text generation. Thomson and Reiter (2020) propose a methodology for human to evaluate the accuracy of the generated texts.

**Sequence-to-Sequence** Seq2Seq paradigm is a general and flexible paradigm that is typically implemented by an encoder-decoder framework. Sutskever et al. (2014) discuss sequence to sequence learning with neural networks. Furthermore, there are some representative architectures that have been proposed such as recurrent neural network (Zaremba et al., 2014) and Transformer (Vaswani et al., 2017). Seq2Seq paradigm can be naturally applied to any task, as long as their input and output can be represented as sequences. Therefore, there have been many attempts to apply Seq2Seq to different tasks. More recently, pre-trained models based on Seq2Seq paradigm (Lewis et al., 2019; Raffel et al., 2019) have proved their power on lots of tasks (McCann et al., 2018; Yan et al., 2021). There has been much work analyzing Seq2Seq models which is always task-specific and based on automatic or human evaluation. For example, Huang et al. (2020) analyze the common models' performance on summarization.

To our knowledge, little work has been done to comprehensively evaluate the performance of Seq2Seq models on data-to-text generation. And much work is based on automatic metrics such as ROUGE or BLEU which can be different from human evaluation as some work (Novikova et al., 2017a; Reiter, 2018; Sulem et al., 2018) shows. Therefore it is meaningful to manually evaluate representative Seq2Seq models on the data-to-text task.

[1] Due to limited computing resources, we didn't evaluate T5-large and BART-large models.

| Dataset | Train Size | Domain | Target Quality | Target Source | Content Selection |
|---|---|---|---|---|---|
| E2E | 50.6K | Restaurants | Clean | Annotator Generated | Partially specified |
| WikiBio | 583K | Biographies | Noisy | Wikipedia | Not specified |
| WebNLG | 25.3K | 15 DBPedia categories | Clean | Annotator Generated | Fully specied |
| ToTTo | 120K | Wikipedia (open-domain) | Clean | Wikipedia (Annotator Revised) | Annotator Highlighted |

Table 1: Summary of data-to-text datasets (Parikh et al., 2020) used in this study

## 3 Models and Datasets

We conduct experiments using five representative Seq2Seq models on four commonly used data-to-text datasets and evaluate the generated texts accordingly[2]. Note that we do not use models that are designed for specific data sets or data structures (Moryossef et al., 2019; Rebuffel et al., 2020; Puduppully and Lapata, 2021), but adopt models that allow inputs of different formats and structures, which brings convenience to comparison on different data sets. Besides, most specific models for data-to-text generation are actually based on these typical Seq2Seq models (Ferreira et al., 2019; Rebuffel et al., 2020), which also proves the rationality of our selection of these models.

### 3.1 Models

We choose to explore and compare Transformer, Pointer Generator, BART and T5's performance on data-to-text generation and explore the role of copy mechanism by comparing Transformer and Pointer Generator, the benefits brought by the pre-training technique by comparing Transformer with T5 and BART, the influence of the different pre-training methods by comparing BART and T5, the power of parameter size by comparing T5-base and T5-small.

**Transformer** Transformer (Vaswani et al., 2017) is widely used in natural language processing and has shown its potential on many tasks. It uses self-attention and multi-head attention which let a model draw from the state at any preceding point along the sequence. The attention layer can access all previous states and weigh them according to a learned measure of relevancy, providing relevant information about far-away tokens. There are also some experiments with Transformer as the baseline model (Zhao et al., 2020) for data-to-text generation. Moreover, many improved models for data-to-text generation are also based on Transformer (Wang et al., 2020; Zhu et al., 2019). Therefore, it is worth and reasonable to explore the performance of Transformer on the data-to-text task.

**Pointer Generator** Pointer Network is first proposed by Vinyals et al. (2015) and See et al. (2017) introduce Pointer Generator based on it. Pointer Generator can generate words from the vocabulary through the generator or copy content from the source through the pointer, which addresses the problem that Seq2Seq models tend to reproduce factual details inaccurately. Copy mechanism is widely used in data-to-text tasks and has achieved great success (Marcheggiani and Perez-Beltrachini, 2018; Rebuffel et al., 2020; Puduppully et al., 2019). Parikh et al. (2020) and lots of other work also use the Pointer Generator as the baseline model. Therefore, the Pointer Generator is a representative model for data-to-text generation. We implement the Pointer Generator based on Transformer so it can take advantage of the copy mechanism.

**BART** BART (Lewis et al., 2019) uses a standard Seq2Seq Transformer architecture with a bidirectional encoder like BERT (Devlin et al., 2018) and a left-to-right decoder like GPT (Radford et al., 2018). The pre-training task involves randomly shuffling the order of the original sentences and a novel in-filling scheme, where spans of text are replaced with a single mask token. With the novel pre-training method and a large number of parameters, BART achieves state-of-the-art on many tasks (Lewis et al., 2020; Siriwardhana et al., 2021). Our results show that BART can perform very well on data-to-text generation too.

**T5** T5 (Raffel et al., 2019) is an encoder-decoder model pre-trained on a multi-task mixture of unsupervised and supervised tasks and for which each task is converted into a text-to-text format whose basic architecture is Transformer. It achieves state-of-the-art on multiple tasks, which shows the power of the large pre-training model and Seq2Seq paradigm. T5-3b (Kale, 2020) obtains the best result on ToTTo dataset. T5-large with a two-step fine-tuning mechanism (Wang et al., 2021)

[2]The codes and annotated data are available at https://github.com/xunjianyin/Seq2SeqOnData2Text

achieves state-of-the-art on WebNLG benchmark. We carry out experiments on T5-small which has 60M parameters and T5-base which has 220M parameters to explore the power of model size.

### 3.2 Datasets

We use the datasets commonly used in data-to-text task in the experiments, including E2E, WebNLG, WikiBio and ToTTo. They have different forms and characteristics, which can give a comprehensive comparison of models. The summary of these data-to-text datasets is shown in Table 1.

**E2E** The input of E2E dataset (Novikova et al., 2017b) is the information about the restaurant, and the output is its natural language description. It consists of more than 50K combinations and the average length of the output text is 8.1 words.

**WikiBio** WikiBio (Lebret et al., 2016) is a personal biography dataset containing more than 70K examples. The input is the infobox from Wikipedia, and the output is the first sentence of the biography. The average length of the output text is 26.1 words.

**WebNLG** The WebNLG challenge (Gardent et al., 2017) consists of mapping sets of RDF triples to text. The latest WebNLG dataset contains more than 40K data-text pairs. The average length of the output text is 22.3 words.

**ToTTo** ToTTo (Parikh et al., 2020) is an open-domain English table-to-text dataset with over 120,000 training examples that proposes a controlled generation task: given a Wikipedia table and a set of highlighted table cells, produce a one-sentence description.

## 4 Evaluation Method

We first evaluate models' performance using automatic metric BLEU (Papineni et al., 2002), and the BLEU scores are comparable to the mainstream research. Then, we use human evaluation similar to PolyTope (Huang et al., 2020) to further analyze and evaluate the performance of the models on different datasets.

BLEU is a precision-based metric for evaluating the quality of generated text and it is widely used by work on data-to-text generation.

Multidimensional Quality Metric (MQM) (Mariana, 2014) is a framework for describing and defining custom translation quality metrics. It defines flexible issue types and a method to generate quality scores. Based on MQM, Huang et al. (2020) introduce an error-oriented fine-grained human evaluation method PolyTope. It defines five issue types about accuracy, three issue types about fluency, syntactic labels and three error severity rules. Note that we do not use the syntactic labels in PolyTope, as they are not the focus of our evaluation in this study. The definitions of our evaluation dimensions are very similar to Huang et al. (2020), but for the sake of the integrity of the paper and more specifically to the task of Data2Text, we still explain them below.

After annotating every generated sentence with these error types and severity, we finally calculate an overall score to evaluate the model's performance.

### 4.1 Issue Type

According to the MQM principle, we define error types in two aspects: accuracy and fluency. Errors related to accuracy mean the generated text is not faithful to the original data or does not reflect the critical information totally from the original data. This type consists of five sub-types:

**Addition** The generated text contains unnecessary and irrelevant fragments from the source data.

**Omission** The key point does not exist in the output.

**Inaccuracy Intrinsic** Terms or concepts appearing in the original data are distorted in the output.

**Inaccuracy Extrinsic** The generated text shows the content which does not exist in the source data.

**Positive-Negative Aspect** The generated text is positive, whereas the source data represents a negative statement and vice versa.

Fluency aspect evaluates the linguistic quality of the generated text, which is a primary natural language requirement. It consists of three sub-types:

**Duplication** Unnecessarily repeat a word or longer part of the text.

**Word Form** Problems related to the form of words, including consistency, part of speech, tense and so on.

| Input | Model's Output |
|---|---|
| **Object**: Austin Texas<br>**Property**: is Part Of **Subject**: Texas | Austin is part of Williamson County Texas<br>where the English is spoken .<br>The largest city in Williamson County is **Georgetown**. |
| **Object**: Texas<br>**Property**: language **Subject**: English language | |
| **Object**: Austin Texas<br>**Property**: is Part Of **Subject**: Williamson County Texas | |
| **Object**: Williamson County Texas<br>**Property**: largest City **Subject**: Round Rock Texas | |
| **Object**: Williamson County Texas<br>**Property**: county Seat **Subject**: Georgetown Texas | |

Table 2: Example output with Inaccuracy Intrinsic and Omission errors. The **Georgetown** is not the largest city but the county Seat so it is the Inaccuracy Intrinsic error. And the generated text do not mention the county Seat so there is an Omission error.

**Word Order** Problems about the order of words in outputs.

One example output with errors on WebNLG dataset is shown in Table 2.

### 4.2 Severity

Severity describes how severe a particular error is. There are three levels: Minor, Major and Critical. Each specific error in the sentence will be allocated a severity. It is decided by the annotator and will be considered as a weight to score the quality of the annotated sentence automatically.

**Minor** Errors that do not affect content availability or understandability. For example, we regard the repetition of function words as an error, but this error will not affect the understanding of the text, so we think this error is Minor.

**Major** Errors that affect content availability or comprehensibility but do not make content unusable. For example, we think additional attributes will not make the content unsuitable for the purpose although it may cause the reader to make additional efforts to understand the intended meaning.

**Critical** Errors that make content unsuitable for use thoroughly. Each error type can make the text completely unusable when it is too severe. For example, when the critical elements in the sentence are missing or too many errors are misleading people's understanding, we think this error is the key.

### 4.3 Calculation

Given original data and generated text, annotators are required to find all errors in the sentence and label them with error types and severity. After the work is done for all samples, the error score of every type and an overall system performance score will be calculated automatically with the below equations:

$$EScore_t = \frac{\sum_{e \in E_t} \alpha_e \times L_e}{word_{count}} \quad (1)$$

$$Score = (1 - \sum_{t \in T} EScore_t) \times 100 \quad (2)$$

where $T$ is the set of error types and $E_t$ is the set of all error segments of type $t$. $\alpha_e$ is the deduction ratio which is set 1:3:7 for the three severity levels: Minor, Major and Critical. $L_e$ is the word length of the error[3]. $word_{count}$ is the total number of words in samples. We can see the highest system performance score can reach 100 if there is no error in the sentences, and it is the higher the better. Through this method, we can get $Score$, an overall evaluation of each model, and error scores $EScore_t$ that indicate each error type's punishment for the overall score.

### 4.4 Human Annotation

After training and testing, we hire five annotators with satisfactory levels in reading from eight candidates. They are all highly educated enough to understand structured data and tables, and their English level is also very high to understand the text. Before formal annotation, we conduct detailed training to make them have a clear understanding of various errors and the severity of PolyTope framework. Examples used in training do not appear in the final annotation. In order to ensure objectivity and impartiality, they know nothing about the name, architecture, BLEU score of the model and dataset in the process of annotation.

[3]Note that we set the length of an Omission error to 1.

| | E2E | WikiBio | WebNLG | ToTTo | Average |
|---|---|---|---|---|---|
| Transformer | 76.88 | 81.31 | 76.32 | 45.41 | 69.98 |
| Pointer-GEN | 86.97 | 82.98 | 78.76 | 54.57 | 75.82 |
| T5-small | 86.04 | 86.28 | 93.92 | 85.44 | 87.92 |
| T5-base | **96.36** | **91.38** | **94.10** | 88.59 | **92.61** |
| BART-base | 91.55 | 86.37 | 93.43 | **90.71** | 90.52 |
| Average | 87.56 | 85.66 | 87.31 | 72.94 | |

Table 3: Human evaluation scores of each model on each dataset (higher means better).

| | E2E | WikiBio | WebNLG | ToTTo |
|---|---|---|---|---|
| Transformer | 56.74 | 43.39 | 27.95 | 33.49 |
| Pointer-GEN | 61.57 | 49.39 | 27.54 | 35.28 |
| T5-small | **62.88** | 49.45 | 55.66 | 45.35 |
| T5-base | 59.96 | 49.12 | **59.48** | **48.91** |
| BART-base | 62.66 | **53.25** | 52.84 | 48.22 |

Table 4: BLEU Scores of each model on every dataset (higher means better).

During testing, annotators are asked to locate every error's position, point out the type of the error, choose the severity of the error and explain the reason. We check their answers and score them. Through the overall performance in the test, we select the best five annotators and ensure all of them really understand our evaluation method and have the ability to do the annotation work.

For each dataset, we select 80 data-text pairs and input them into each model respectively. There are four datasets and five models, so we have 1600 texts to annotate. Each text is annotated by two different annotators respectively and if the difference of their error scores is too large, the text will be abandoned and a new text will be selected to join the evaluation. They are not allowed to communicate with each other in the annotation process. They can choose to abandon the texts that confuse them, and these texts will be replaced by candidate texts. Each annotator must label all the five outputs generated by five models of one input sequence at a time to keep equality. In general, we strive to balance the fairness and quality of the evaluation.

## 5 Result Analysis

We evaluate the five models mentioned above on four datasets using the above metrics. The overall human evaluation score and BLEU score of each model on each dataset are shown in Table 3 and Table 4, respectively. The detailed error scores of different error types are shown in Table 5. We can compare the performance of the models to see the influence of the pre-training technique, the copy technique and the mode size. Comparing the results on different datasets using the same model, we can discover how the structured data input influences the performance of the Seq2Seq models. Moreover, we can also analyze the detailed error scores to find out the weakness and advantages of specific models.

### 5.1 Copy Mechanism

Through comparing the results of Pointer Generator and Transformer on all datasets, we can see that the copy mechanism has an noticeable effect on the improvement of the results. It improves the generation performance on all the datasets. Particularly, it reduces the Inaccuracy Intrinsic error score by about 3 or 4 points on three datasets (E2E, WebNLG and ToTTo), as shown in Table 5. It is easy to understand because using copy mechanism, the model can generate words from the vocabulary through the generator or copy content from the source through the pointer. Pointer Generator with copy mechanism reduces almost all types of errors compared with vanilla Transformer such as Duplication error. The reason may be that the copy mechanism can interpolate vocabulary level probability with copy probability, reducing reliance on previous outputs.

We can observe that the improvement of Pointer Generator over Transformer is the largest on ToTTo dataset. This may be related to ToTTo's need to pay more attention to the highlighted part of the input sequence, which emphasizes controllability.

Nevertheless, it is interesting that Addition error is increased slightly compared with Transformer. The likely reason may be that the auto-regressive decoder tends to copy longer sequences from the source and it is hard to interrupt the copy action.

| dataset | model | Addition | Duplication | Extrinsic | Intrinsic | Omission | Positive-Negative Aspect | Word Form | Word Order |
|---|---|---|---|---|---|---|---|---|---|
| E2E | Transformer | 2.52 | 0 | 5.46 | 7.14 | 5.97 | 0 | 0 | 2 |
| | ptr-gen | 2.41 | 0.33 | 1.66 | 3.56 | 2.45 | 0 | 0.92 | 1.66 |
| | T5-small | 0.99 | 0 | 0 | 5.18 | 5.06 | 1.75 | 0 | 0.95 |
| | T5-base | 0 | 0 | 0 | 1.6 | 1.11 | 0 | 0 | 0.91 |
| | BART-base | 0.81 | 0 | 0.81 | 1.53 | 3.73 | 0 | 0 | 1.53 |
| WikiBio | Transformer | 0.38 | 1.53 | 4.52 | 2.73 | 8.82 | 0.67 | 0 | 0 |
| | ptr-gen | 1.47 | 0.86 | 3.51 | 2.97 | 7.70 | 0.49 | 0 | 0 |
| | T5-small | 0.69 | 0 | 1.22 | 3.35 | 8.44 | 0 | 0 | 0 |
| | T5-base | 0 | 0 | 1.32 | 2.17 | 4.83 | 0 | 0.28 | 0 |
| | BART-base | 0.15 | 0 | 1.15 | 2.70 | 9.61 | 0 | 0 | 0 |
| WebNLG | Transformer | 1.02 | 2.84 | 1.89 | 10.44 | 7.03 | 0 | 0.44 | 0 |
| | ptr-gen | 3.90 | 2.69 | 0 | 6.38 | 7.27 | 0 | 0 | 0.97 |
| | T5-small | 0 | 0.69 | 0 | 4.56 | 0.81 | 0 | 0 | 0 |
| | T5-base | 0 | 0 | 0.34 | 3.50 | 1.49 | 0.54 | 0 | 0 |
| | BART-base | 0 | 0 | 0.44 | 4.45 | 1.66 | 0 | 0 | 0 |
| ToTTo | Transformer | 4.38 | 2.38 | 11.03 | 17.02 | 19.74 | 0 | 0 | 0 |
| | ptr-gen | 11.01 | 1.31 | 9.11 | 13.47 | 7.48 | 0 | 3.01 | 0 |
| | T5-small | 0 | 0 | 0 | 5.36 | 9.19 | 0 | 0 | 0 |
| | T5-base | 0 | 0 | 1.86 | 4.38 | 4.38 | 0 | 0 | 0.76 |
| | BART-base | 0 | 0 | 1.79 | 2.41 | 2.66 | 0 | 0 | 2.41 |

Table 5: Error score of each error type for each model on 80 data-text pairs of every dataset. The results are scored based on manual evaluation and retained to two decimal places. Lower means better. Errors may be approximated to 0 because there are too few errors.

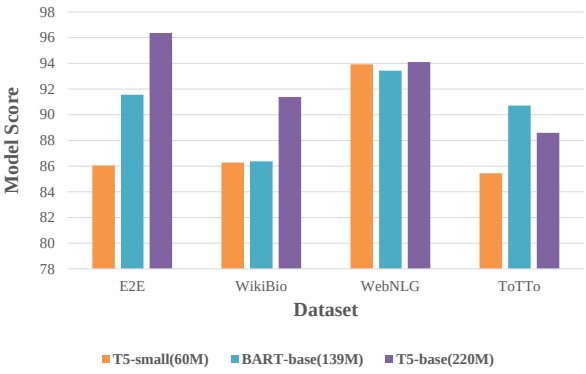


Figure 1: Comparison results of pre-trained models with different numbers of parameters (higher means better).

## 5.2 Pre-training

In Table 3, we can see almost all the pre-training models outperform the non-pre-training models by a large margin among all the datasets except E2E dataset which may be too simple to evaluate the ability of models. The reason why the pre-training models can achieve better scores may be that they have learned helpful knowledge from lots of raw texts. And the pre-training method also helps the models become more powerful. BART and T5 are both pre-trained on tasks where spans of text are replaced by masked tokens. The models must learn to reconstruct the original document. According to the average scores of all the datasets, we can say that T5-base may be the best Seq2Seq model among our experimented models and BART-base is not far behind. And the models achieve the highest score on different datasets: BART-base is the best on ToTTo and T5-base is the best on the other datasets relatively.

## 5.3 Model Size

It is evident that the parameter quantity is the critical factor to the pre-trained model's performance. BART-base has 139M parameters, T5-base has 220M parameters and T5-small has 60M parameters only. With the same architecture and same pre-training method, T5-base totally outperforms T5-small. Due to pre-training methods and other factors, T5-base and BART-base achieve the best results on different datasets. But on average, T5-base is the best. The relation between model size and the performance on different datasets is shown in Figure 1. The only exception mentioned above is ToTTo, where BART-base achieves the best results. One of the likely reasons is the pre-training strategy of BART which helps it have better denoising and reconstruction ability. Another reason will be mentioned in section 5.4.

## 5.4 Dataset

We can compare the difficulty level of the datasets by the average and the highest scores of all models. In Table 3, the ToTTo dataset has the lowest average score of 72.9. And the highest score on it achieved by BART-base is 90.7 which is also the lowest among all the datasets. ToTTo is made as a controlled generation task that given a Wikipedia

| | Addition | Duplication | Extrinsic | Intrinsic |
|---|---|---|---|---|
| Average Scores | 1.48 | 0.63 | 2.30 | 5.25 |
| | Omission | Positive-Negative Aspect | Word Form | Word Order |
| Average Scores | 5.97 | 0.17 | 0.23 | 0.56 |

Table 6: Average error score of each error type across all models and datasets (lower means better).

table and a set of highlighted table cells, the model needs produce a one-sentence description of the highlighted part. It is much more complicated than other datasets describing all the given structured data. Maybe it is a bit confusing for models to find out what actually should be noticed, although the scores of the pre-training models are still very high. And the gap between pre-training models and non-pre-training models is the biggest on ToTTo among all datasets which indicates that the simple non-pre-training models can not handle the complex controlled generation very well. Of course the quantity of the data-text pairs and the length of the input and output sequence also influence the models' performance.

### 5.5 Error Types

Table 6 shows the average error scores of each error type across all models and datasets. From Table 5 and Table 6, we can find that different types of errors have different effects on the performance of the models. We can find that Omission Error is the most frequent and severe error and its error score is almost up to 6. The likely reason is that the input sequence is too long, so it is hard to encode all its meaning. So the models tend to omit some information from the input. And Inaccuracy Intrinsic Error and Inaccuracy Extrinsic Error also can not be ignored which are 5.25 and 2.31, respectively. From the perspective of the pre-training model, it may be because they learn too much from the raw texts on pre-training stage and the knowledge lets them tend to generate inaccurate texts.

It is excited that all the models perform very well in terms of fluency. The errors of Duplication, Word Form and Word Order are very sporadic. This shows the Seq2Seq models can generate fluent text with the structured input.

## 6 BLEU or Human Evaluation?

We can see that the overall trend of the BLEU score is consistent with human evaluation, which can basically reflect the overall performance of the model. And many conclusions we made above can also be proved by the BLEU score. For example, the biggest pre-training model T5-base achieves the highest BLEU score too among the selected models, Pointer Generator with copy mechanism still performs better than Transformer and ToTTo is still the most difficult dataset.

Although our primary goal is not to promote a human evaluation metric, our dataset with human annotations gives us a testbed to analyze the correlations and differences between automatic and human metrics. There have been a lot of discussions in the community about the unreliability of BLEU metric. Sulem et al. (2018) recommend not using BLEU on text simplification. They found that BLEU scores can neither reflect grammar nor the meaning of preservation. Novikova et al. (2017a) show that BLEU and some other commonly used indicators are not well consistent with human judgment when evaluating NLG tasks.

We compute the Pearson correlation coefficients between BLEU score and manual evaluation in terms of *Accuracy* and *Fluency*. We categorize the error types into accuracy and fluency aspects according to the definition in Section 4.1, and use Equation 2 to calculate Accuracy score and Fluency score respectively. The Pearson correlation coefficient between BLEU score and *Accuracy* is 0.61 and in *Fluency* aspect is 0.08. There is a huge gap between them and we can see that BLEU can evaluate *Accuracy* to a certain extent and it is poor at *Fluency*. Moreover, the BLEU metric is too coarse-grained to reveal the model's specific problems, which enlighten us on how to improve the model. Our result is consistent with views of other work.

## 7 Conclusion

We empirically compared five representative Seq2Seq models on the data-to-text task using a fine-grained set of human evaluation metrics based on MQM. We aim to make a systematic and comprehensive evaluation and analysis on end-to-end Seq2Seq models for the data-to-text task. We analyze the effect of milestone techniques such as copy

and pre-training, the influence of the dataset and model size and the models' performance in terms of different types of errors. Our evaluation shows that pre-trained models can generate quite good texts. But there is still much room for improvement in this task. Furthermore, the improvement of specific errors such as Omission Error and Inaccuracy Intrinsic Error is also worth exploring in the future.

## Acknowledgements

This work was supported by National Key R&D Program of China (No.2018YFB1005100),Bejing Academy of Artificial Intelligence (BAAI) and State Key Laboratory of Media Convergence Production Technology and Systems. We would like to appreciate the anonymous reviewers for their helpful comments. Xiaojun Wan is the corresponding author.

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Thiago Castro Ferreira, Chris van der Lee, Emiel Van Miltenburg, and Emiel Krahmer. 2019. Neural data-to-text generation: A comparison between pipeline and end-to-end architectures. *arXiv preprint arXiv:1908.09022*.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for nlg micro-planning. In *55th annual meeting of the Association for Computational Linguistics (ACL)*.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170.

Dandan Huang, Leyang Cui, Sen Yang, Guangsheng Bao, Kun Wang, Jun Xie, and Yue Zhang. 2020. What have we achieved on text summarization? *arXiv preprint arXiv:2010.04529*.

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

Mihir Kale and Abhinav Rastogi. 2020. Template guided text generation for task-oriented dialogue. *arXiv preprint arXiv:2004.15006*.

Karen Kukich. 1983. Design of a knowledge-based report generator. In *21st Annual Meeting of the Association for Computational Linguistics*, pages 145–150.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. *arXiv preprint arXiv:1603.07771*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2005.11401*.

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

Valerie R Mariana. 2014. *The Multidimensional Quality Metric (MQM) framework: A new framework for translation quality assessment*. Brigham Young University.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2015. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. *arXiv preprint arXiv:1509.00838*.

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

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017a. Why we need new evaluation metrics for nlg. *arXiv preprint arXiv:1707.06875*.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017b. The e2e dataset: New challenges for end-to-end generation. *arXiv preprint arXiv:1706.09254*.

Jekaterina Novikova and Verena Rieser. 2016. The analogue challenge: Non aligned language generation. In *Proceedings of the 9th International Natural Language Generation conference*, pages 168–170.

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.

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. *arXiv preprint arXiv:2004.14373*.

Laura Perez-Beltrachini and Claire Gardent. 2017. Analysing data-to-text generation benchmarks. *arXiv preprint arXiv:1705.03802*.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-text generation with content selection and planning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 6908–6915.

Ratish Puduppully and Mirella Lapata. 2021. Data-to-text generation with macro planning. *Transactions of the Association for Computational Linguistics*, 9:510–527.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

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

Clément Rebuffel, Laure Soulier, Geoffrey Scoutheeten, and Patrick Gallinari. 2020. A hierarchical model for data-to-text generation. *Advances in Information Retrieval*, 12035:65.

Ehud Reiter. 2018. A structured review of the validity of bleu. *Computational Linguistics*, 44(3):393–401.

Ehud Reiter and Robert Dale. 2000. Building natural generation systems. *Studies in Natural Language Processing. Cambridge University Press*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Shamane Siriwardhana, Rivindu Weerasekera, Elliott Wen, and Suranga Nanayakkara. 2021. Fine-tune the entire rag architecture (including dpr retriever) for question-answering. *arXiv preprint arXiv:2106.11517*.

Elior Sulem, Omri Abend, and Ari Rappoport. 2018. Bleu is not suitable for the evaluation of text simplification. *arXiv preprint arXiv:1810.05995*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Craig Thomson and Ehud Reiter. 2020. A gold standard methodology for evaluating accuracy in data-to-text systems. *arXiv preprint arXiv:2011.03992*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. *arXiv preprint arXiv:1506.03134*.

Qingyun Wang, Semih Yavuz, Victoria Lin, Heng Ji, and Nazneen Rajani. 2021. Stage-wise fine-tuning for graph-to-text generation. *arXiv preprint arXiv:2105.08021*.

Tianming Wang, Xiaojun Wan, and Hanqi Jin. 2020. Amr-to-text generation with graph transformer. *Transactions of the Association for Computational Linguistics*, 8:19–33.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, and Steve Young. 2016. Multi-domain neural network language generation for spoken dialogue systems. In *Proceedings of the 2016 Conference on North American Chapter of the Association for Computational Linguistics (NAACL)*. Association for Computational Linguistics.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2017a. Challenges in data-to-document generation. *arXiv preprint arXiv:1707.08052*.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2017b. Challenges in data-to-document generation. *arXiv preprint arXiv:1707.08052*.

Hang Yan, Junqi Dai, Xipeng Qiu, Zheng Zhang, et al. 2021. A unified generative framework for aspect-based sentiment analysis. *arXiv preprint arXiv:2106.04300*.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Chao Zhao, Marilyn Walker, and Snigdha Chaturvedi. 2020. Bridging the structural gap between encoding and decoding for data-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2481–2491.

Jie Zhu, Junhui Li, Muhua Zhu, Longhua Qian, Min Zhang, and Guodong Zhou. 2019. Modeling graph structure in transformer for better AMR-to-text generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5459–5468, Hong Kong, China. Association for Computational Linguistics.