Search and Learn: Improving Semantic Coverage for Data-to-Text Generation

Shailza Jolly
TU Kaiserslautern, Germany
DFKI GmbH, Germany
shailza.jolly@dfki.de

Andreas Dengel
TU Kaiserslautern,
Germany
DFKI GmbH, Germany
andreas.dengel@dfki.de

Lili Mou
University of Alberta;
Alberta Machine
Intelligence Institute
(Amii)
doublepower.mou@gmail.com

Abstract

Data-to-text generation aims to generate text description based on input data. Modern approaches are largely based on neural networks, which often require massive parallel data for training. Recently, researchers address few-shot data-to-text generation by fine-tuning pre-trained language models. In our work, we observe that such few-shot models suffer from the problem of *low semantic coverage*, i.e., important input slots are missing in the generated text. We therefore propose a search-and-learning approach that inserts the missing slots in a greedy manner and learns from the search results. Results show that our model achieves high performance on E2E and WikiBio datasets. Especially, we cover 98.5% of input slots on the E2E dataset, largely alleviating the low coverage problem.

1 Introduction

Data-to-text generation is a task converting structured information into human-readable text descriptions, illustrated in Figure 1. Data-to-text generation has gained much attention in the field of natural language processing, with applications to restaurant descriptions (Novikova et al., 2017), biographies (Lebret et al., 2016), and weather forecasts (Liang et al., 2009).

Traditional approaches to natural language generation (NLG) use handcrafted rules with statistics (Langkilde and Knight, 1998; Stent et al., 2004; Rieser and Lemon, 2009), usually lacking flexibility and diversity of the outputs.

Recently, data-to-text generation is generally accomplished by modern neural networks, such as sequence-to-sequence recurrent neural networks (Lebret et al., 2016; Liu et al., 2018). These models use massive parallel training datasets, for example, 42K table–text training pairs in the E2E dataset (Novikova et al., 2017). This data-hungry nature of neural models makes data-to-text generation an expensive and time-consuming affair and restricts its real-world applications.

Chen et al. (2020b) present a few-shot learning approach for data-to-text generation by fine-tuning pre-trained language models (LMs) with a copy mechanism. We observe that, in the few-shot setting, the fine-tuned LMs fail to fully learn the correspondence between input and output. They suffer from the problem of *low semantic coverage*, that is, important information slots are often missing in the generated text.

In our work, we propose a search-and-learning (S&L) approach to address the low coverage problem for few-shot data-to-text generation. We first fine-tune the pretrained T5 language model (Raffel et al., 2020), similar to previous work (Chen et al., 2020b). To address the low coverage problem, we iteratively insert a missing slot into the generated sentence at all possible positions, and pick the most appropriate candidate sentence based on T5 probability. This can be thought of as greedy search that finds a sentence containing all slots. Inspired by Li et al. (2020), we then treat the search results as pseudo-groundtruth and further fine-tune our T5 language model. In this way, our model achieves high performance, especially with a high coverage of the input slots. Also, our model is efficient for generating sentences and does not increase the inference complexity, compared with search-based inference (Liu et al., 2020).

In summary, our main contributions include:

- We address the *low semantic coverage* problem in few-shot data-to-text generation.
- We propose a search-and-learning approach by inserting missing slots and learning from the search results.
- We conduct extensive experiments on E2E and WikiBio datasets. Our model outperforms previous few-shot models in various metrics, largely closing the gap between few-shot and fully supervised learning.

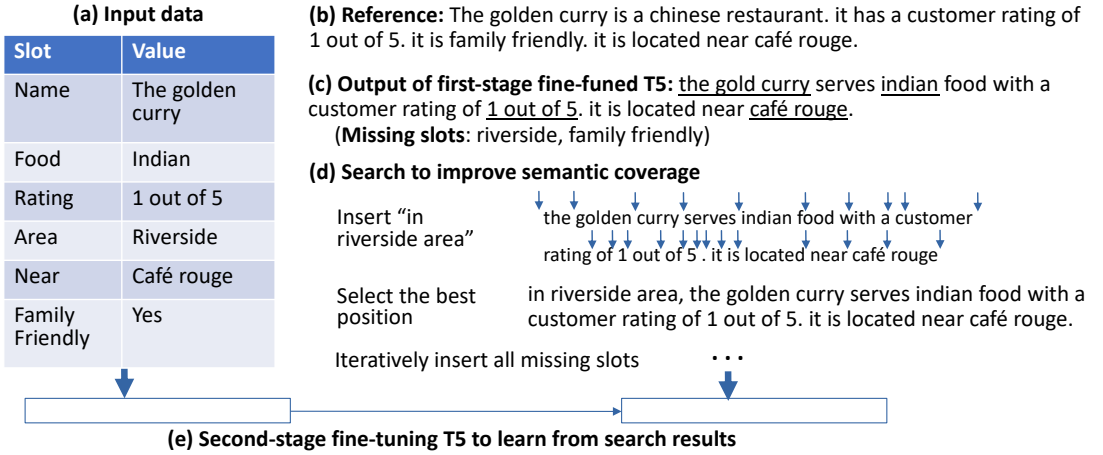

Figure 1: An example for data-to-text generation and our proposed approach.

2 Related Work

2.1 Data-to-Text Generation

Generating human-understandable sentences from tabular data, commonly known as data-to-text generation, is a persistent problem from early NLP research. Traditional work typically follows a pipeline approach of sentence/content planning and surface realization (Dale and Reiter, 1997), using hand-engineered rules (Kukich, 1983; McKeown, 1992) and statistical induction (Liang et al., 2009; Koncel-Kedziorski et al., 2014). However, the generated text by such systems usually lacks flexibility and diversity.

With the rise of deep learning, neural models have become a prevailing approach to data-to-text generation (Lebret et al., 2016; Liu et al., 2018; Wiseman et al., 2018; Liu et al., 2019). Typically, these systems require massive parallel data for training the text generator.

Very recently, Chen et al. (2020b) addressed few-shot learning for data-to-text generation, where they assume a very small parallel corpus is available. They propose to fine-tune pre-trained language models (LMs) with a copy mechanism. We observe that, even such fine-tuned LMs generate fluent sentences, they suffer from the problem of low semantic coverage.

In fact, Dhingra et al. (2019) have pointed out the low semantic coverage problem of human-written references. They propose a new metric for better evaluating data-to-text models. In this paper, we emphasize the low coverage problem of text generators, and propose a search-and-learning approach to overcome it.

Gong et al. (2020) improve the fidelity of data-to-text generators by table reconstruction and content matching along with fine-tuning GPT-2. Our preliminary analysis during development suggests that fine-tuned T5 does not generate wrong information, but may miss important input slots.

2.2 Search-Based Text Generation

Previous work has addressed unsupervised text generation by various search approaches, such as simulated annealing (Liu et al., 2020) and hill-climbing (Schumann et al., 2020). The basic idea is to define a heuristic objective function (typically involving language fluency, semantic coherence, and other task-specific scores) and generate text by word-level editing towards the objective. Li et al. (2020) further propose a search-and-learning approach to improve performance and inference efficiency.

Our paper adopts the framework of Li et al. (2020), but differs from previous search-based approaches in several significant ways: 1) We address the few-shot setting. Instead of a heuristically defined objective, we use a fine-tuned language model to evaluate candidate sentences. 2) The goal of our search is for a higher semantic coverage, rather than a generic fluent sentence. Our search space is relatively simpler than the entire sentence space, and thus, the main focus of our work is not the search algorithm. We adopt greedy search over multiple missing slots, which turns out to work well empirically. To the best of our knowledge, we are the first to address few-shot data-to-text generation by search and learning.

Other work learns to perform word editing in a supervised way (Dong et al., 2019), or treat rule-based editing as input (Li et al., 2018). We instead

perform editing to search for high semantic coverage and further learn from the edit results.

3 Problem Formulation

Data-to-text generation aims to generate a natural language description for some structured input data; we consider a common setting, where the input is tabular data. For one data sample, the input table is a set of slot name–value pairs, denoted by $\mathbf{T} = \{(s_i, v_i)\}_{i=1}^{S}$, where s_i is the name of the ith slot, v_i is the value, and S is the number of slots. The output is a sentence $\mathbf{y} = (y_1, y_2, \cdots, y_m)$ that describes the given input $\mathbf{T}$. Notice that the table $\mathbf{T}$ is different for every data sample, but we omit the sample index for simplicity.

In this work, we consider the few-shot setting, where we have a small parallel corpus $\mathcal{D}_p = \{(\mathbf{T}^{(m)}, \mathbf{y}^{(m)})\}_{m=1}^{M}$ and another small unlabeled corpus $\mathcal{D}_u = \{\mathbf{T}_u^{(n)}\}_{n=1}^{N}$, where $\mathbf{T}_u^{(n)}$ is a different table than $\mathbf{T}^{(m)}$. Here, both M and N are small numbers in our few-shot setting.

Few-shot learning is important to NLG, as it saves human annotation labors and also helps to alleviate the cold-start problem of an NLG task. In our work, we assume a small unlabeled corpus $\mathcal{D}_u$ is available in addition to $\mathcal{D}_p$. This is a realistic setting for few-shot learning, because unlabeled data are easier to obtain than labeled pairs with human-written sentences, and sometimes $\mathcal{D}_u$ may be synthesized by recombining the slots of $\mathcal{D}_p$ for data-to-text generation.

4 Proposed Model

In our approach, we first fine-tune a pre-trained language model (LM) for conditional text generation based on an input table (Section 4.1). This allows us to generate fluent sentences in the few-shot setting, due to the large model capacity and extensive pre-training of the LM.

However, fine-tuned LMs may not fully learn the correspondence between input slots and output text, and have the problem of *low semantic coverage*. Thus, we iteratively insert a missing slot into the generated sentence in a greedy manner, so as to improve the semantic coverage (Section 4.2). Finally, these search results are treated as pseudo-groundtruth for further fine-tuning our LM, which not only improves inference efficiency, but also yields better sentences (Section 4.3).

4.1 First-Stage Fine-Tuning T5

We use the T5 model (Raffel et al., 2020) for data-to-text conditional generation. T5 is a text-to-text Transformer model (Vaswani et al., 2017), which is pre-trained on multiple NLP tasks and achieves state-of-the-art performance on question answering, document summarization, sentiment classification, etc. In fact, T5 can be thought of as an alternative for BERT (Devlin et al., 2019) and GPT-2 (Radford et al., 2019).

It is worth noting that T5 is never pre-trained on any tasks related to data-to-text generation. Therefore, our experiments are indeed in the few-shot setting even if we use pre-trained T5.

We linearized the input table by concatenating all slots in the format of "`name[value]`". In other words, a special token "`[`" separates the name and value of a slot, and another special token "`]`" separates different slots.

In our few-shot setting, we fine-tune T5 using several hundred data–text pairs, which is considerably smaller than a usual NLG training set. The model learns to estimate the conditional probability $P(\mathbf{y}|\mathbf{T})$ in an auto-regressive way:

$$P(\mathbf{y}|\mathbf{T};\boldsymbol{\theta}) = \prod_{i=1}^{n} P(y_i|\mathbf{y}_{<i}, \mathbf{T};\boldsymbol{\theta}), \qquad (1)$$

where $\mathbf{y}$ is the output with length n, $\mathbf{T}$ is the input table, and $\boldsymbol{\theta}$ represents model parameters.

We fine-tune T5 with the cross-entropy loss

$$J(\boldsymbol{\theta}) = -\log P(\mathbf{y}|\mathbf{T};\boldsymbol{\theta}) \qquad (2)$$

4.2 Search to Improve Semantic Coverage

We observe that T5 indeed generates fluent sentences, but it has a low coverage of the input slots. In Figure 1c, for example, the slots "riverside" and "family friendly" are not mentioned in the generated text, although they are present in the groundtruth.

The low semantic coverage is because the few-shot parallel corpus cannot fully support T5 learning the correspondence between input and output. This is evidenced by analyzing the coverage percentage and the training size: T5 fine-tuned on 1% of E2E data has a coverage of 84.46% input slots, whereas it has a coverage of 97.74% if fine-tuned on the whole dataset.

To this end, we propose a simple yet effective search approach that explicitly inserts the missing slots into the generated text.

We start by checking the occurrence of each slot value v_i in T5's output. If a slot value v_i does not appear in the output, we insert a phrase $\tilde{v}_i$ based on manually designed templates. For example, if the slot "`area[riverside]`" is missing, we insert the phrase $\tilde{v}_i =$ "`in riverside area`". The E2E dataset has a boolean slot "`familyFriendly[yes/no]`", and we design the phrase template as either "`family friendly`" or "`not family friendly`". The complete list of our templates is shown in Appendix A.

For every missing slot, we determine the most appropriate position for inserting the slot template. This is given by an enumeration of all possible positions within a sentence, and we select the candidate that has the highest T5 probability $p(\mathbf{y}|\mathbf{T})$ as fine-tuned by Eqn (1), shown in Figure 1.

This process is repeated for all missing slots, and every missing slot is inserted greedily according to the above criterion. Our approach can be thought of as an optimization towards

$$\hat{\mathbf{y}}_{\text{search}} = \text{argmax } p(\mathbf{y}|\mathbf{T}) \quad \text{subject to } v_i \in \mathbf{y}, \forall i \tag{3}$$

Specifically, we optimize the T5 conditional probability for data-to-text generation by starting from an *infeasible* solution (i.e., an output that violates the constraint). We then project the solution into the *feasible* set by satisfying each constraint in a greedy fashion.

Our search method is inspired by recent development of search-based unsupervised text generation, such us simulated annealing for paraphrasing (Liu et al., 2020) and hill-climbing for summarization (Schumann et al., 2020). However, our search effort is mainly devoted to projecting an infeasible solution to the feasible set, instead of searching for a generic sentence that maximizes the objective.

Our approach also resembles rule-based text generation systems in the early age (Langkilde and Knight, 1998; Stent et al., 2004), as we also design rules and templates. However, our work differs from them significantly, as we use rules only for revision not for generation. Further, we will have a learning component that learns from the search results, introduced in the next subsection.

4.3 Second-Stage Fine-Tuning T5 with Search Results

The search approach in Section 4.2 ensures a high semantic coverage in the output sentence, but it has two major drawbacks: 1) the edited sentence may not be fluent due to the fixed template, and 2) it has a low inference efficiency as we have to evaluate multiple candidate outputs given a data sample. To address the above drawbacks, we further fine-tune T5 that learns from the search results, inspired by Li et al. (2020).

In our few-shot setting, we assume there is small unlabeled corpus $\mathcal{D}_u$ containing input tables only. In practice, $\mathcal{D}_u$ can either be obtained inexpensively or can be synthesized by recombining the table slots and values in $\mathcal{D}_p$.

For a given input table $\mathbf{T}_u^{(i)} \in \mathcal{D}_u$, we use T5 to generate a candidate output (Section 4.1) and perform search for higher semantic coverage (Section 4.2). The search result is treated as a pseudo-groundtruth, denoted by $\hat{\mathbf{y}}_{\text{search}}^{(i)}$. It is mixed with the original parallel corpus for further fine-tuning T5. In other words, our dataset becomes $\mathcal{D}_p \bigcup \{(\mathbf{T}_u^{(i)}, \hat{\mathbf{y}}_{\text{search}}^{(i)}) : \mathbf{T}_u^{(i)} \in \mathcal{D}_u\}$. T5 is fine-tuned by the same cross-entropy loss as Equation (2). Details are not repeated.

4.4 Inference

For inference on the test set, we only use the two-stage fine-tuned T5 (i.e., fine-tuned on search results in Section 4.3) to predict the output. We do not have the search procedure during inference.

In this way, our inference efficiency is improved than the search approach, because we do not have to evaluate multiple candidate sentences during prediction. More importantly, we leverage the power of pre-trained language models, and are able to generate more fluent sentences than the search itself.

Compared with one-stage fine-tuning (Section 4.1), T5 this time is explicitly trained with pseudo-groundtruth that has high semantic coverage. We will show in Section 5.1 that, after search and learning, T5 achieves near-perfect semantic coverage on the E2E dataset.

#	Model	#Train	BLEU	NIST	METEOR	RougeL	CIDEr	PARENT (P/R/F1)	PPL	AvgLen	Coverage
1	TGEN	p:42K	65.93	8.61	44.83	68.50	2.23	–	–	–	–
2	SLUG	p:42K	66.19	8.61	44.54	67.72	–	–	–	–	–
3	S_1^R	p:42K	**68.60**	8.73	45.25	70.82	2.37	–	–	–	–
4	T5	p:42K	67.59	**8.81**	**45.17**	**70.44**	**2.33**	67.40 / 61.75 / 63.43	154.49	23.58	97.74%
5	T5	p:2100	62.45	8.30	44.10	67.15	2.17	64.25 / 61.69 / 62.00	136.54	24.82	96.69%
6	T5	p:420	**61.72**	7.96	40.52	65.61	1.96	65.63 / 57.25 / 60.10	**141.61**	21.97	84.46%
7	T5 self-train	p:420, u:1680	60.83	7.74	39.85	66.36	1.95	66.60 / 57.31 / 60.63	154.74	21.50	83.21%
8	T5 S&L	p:420, u:1680	60.70	**8.13**	**43.60**	**65.84**	**2.12**	**66.97 / 63.63 / 64.29**	160.40	25.01	**98.50%**

Table 1: Test results on E2E. "p:" and "u:" denote the number of parallel and unlabeled training samples, respectively. Baseline results are quoted from original papers (Novikova et al., 2017; Juraska et al., 2018; Shen et al., 2019). All results of fine-tuning T5 are obtained by our experiments. Based on the evidence in Dhingra et al. (2019), we consider PARENT as our main metrics.

5 Experiments

5.1 Experiment I: E2E Dataset

Dataset. In this experiment, we used the E2E dataset[1] (Novikova et al., 2017) to evaluate our approach. E2E is a crowdsourced dataset for data-to-text generation and contains more than 50K table–text pairs for the restaurant domain. For each data sample, the input contains 3–8 slots, and the reference contains one or a few sentences as the output. We followed the standard train/val/test split.

Implementation details. We used the T5-small model (Raffel et al., 2020), which comprises 6 layers in the encoder and the decoder. We trained models using the AdamW (Loshchilov and Hutter, 2018) optimizer, with an initial learning rate of 3e-4 and a batch size of 64.

Evaluation metrics. We used the standard evaluation scripts accompanied with the E2E dataset (Novikova et al., 2017), including BLEU (Papineni et al., 2002), NIST (Doddington, 2002), METEOR (Lavie and Agarwal, 2007), ROUGE-L (Lin, 2004), and CIDEr (Vedantam et al., 2015).

Recently, Dhingra et al. (2019) observe that BLEU correlates poorly to human satisfaction for data-to-text generation, as we have diverse references. They propose a set of PARENT metrics (including precision, recall, and the F-score) against both the references and the input data. They show PARENT metrics have high correlation with human judgment. Based on such evidence, we do not rely on BLEU for evaluating our approaches, and we adopt PARENT as the main metric.

In addition, we use GPT-2 perplexity (without fine-tuning) to estimate the fluency of generated text, and present the average sentence length (AvgLen) for reference. We also computed semantic coverage ratio, which is the fraction of input slots that appear verbatim in the output. This requirement appears to be strict, but is actually a good approximation, because most slots contain only one or a few words and some slots are proper nouns that should not be changed. We also conducted human evaluation on a randomly selected subset of test samples, and the coverage percentage (Table 3) is close to this automatic metric.

Results. Table 1 shows the results on the E2E dataset. We consider a few-shot setting, where we have 1% parallel samples as $\mathcal{D}_p$, and another 4% samples as $\mathcal{D}_u$ with input tables only.

Before few-shot learning, we fine-tuned T5 with 100% samples (Line 4) and 5% samples (Line 5), respectively, being an "upper bound" performance of our few-shot learning. We see that, with the entire dataset, fine-tuning T5 achieves similar scores to previous state-of-the-art models, including TGEN (Novikova et al., 2017), SLUG (Juraska et al., 2018), and the S_1^R model (Shen et al., 2019). This shows that the use of T5 sets up a solid foundation for our study.

We started few-shot learning by directly fine-tuning T5 on the small parallel training set. We observe that the performance worsens in all metrics (comparing Lines 4–6), especially the coverage drops quickly from 97.74% to 84.46%.

We would like to see if a small unlabeled dataset $\mathcal{D}_u$ that contains tables only could help the performance. We experimented self-training (Zhu and Goldberg, 2009), which is a common strategy for semi-supervised machine learning. In this competing method, we first fine-tune T5 on the parallel corpus $\mathcal{D}_p$, and use it to predict the output on $\mathcal{D}_u$. The predicted sentences are treated as pseudo-groundtruth for further fine-tuning. Unfortunately,

[1] http://www.macs.hw.ac.uk/InteractionLab/E2E/

we observe from Lines 6 and 7 that such strategy does not help the performance much.

Finally, we applied our S&L approach, shown in Line 8. Results show that our approach achieves the highest performance in the few-shot setting in terms of most metrics. Especially, we achieve a 98.50% coverage of input slots, which mostly solves the *low coverage* problem.

Based on the numerical results, it appears that our model generate less fluent sentences, given by high perplexity (PPL) scores.[2] However, we notice that our sentences are longer and contain more input slots, which are oftentimes very specific information such as the restaurant name (in the E2E dataset) as a proper noun. Therefore, it is understandable that our PPL is slightly higher, but in general, all models are in the same ballpark in terms of fluency. This will be further analyzed by human evaluation (Table 3).

Generally, our S&L approach (Line 8) achieves comparable results to T5 trained with 4 times more parallel data (Line 5) in several metrics, such as METEOR and CIDEr. In terms of PARENT metrics that are specifically designed for data-to-text generation, we observe our S&L approach outperforms Line 5 with a reasonable margin. It even achieves close PARENT scores to the fully-supervised setting (Line 4). Since we only used 1% parallel data and 4% unlabeled input, this convincingly show that our approach achieves high performance for few-shot data-to-text generation.

5.2 Experiment II: WikiBio Dataset

Dataset. We further evaluate our approach on the Humans domain of the WikiBio dataset[3] (Lebret et al., 2016). WikiBio contains 700K English biographies from Wikipedia, associated with a tabular infobox. For each biography, the first sentence of the article is treated as the reference.

In our few-shot setting, we used 100 parallel samples as the training set $\mathcal{D}_p$, following one of the settings in Chen et al. (2020b). In accordance with our assumption, we included another 400 samples of unlabeled input tables as $\mathcal{D}_u$. When comparing with Chen et al. (2020b), however, we did not use $\mathcal{D}_u$, but synthesized 400 samples by recombining the table slots in $\mathcal{D}_p$. This sets up a fair comparison, as we did not include any new data. We validated our approach on 1000 samples, and tested it on the standard split.

Implementation details. We used the T5-base model, which consists of a 12-layer Transformer encoder and decoder. This sets up a fair comparison with the prior work for few-shot data-to-text generation (Chen et al., 2020b), which uses a 12-layer GPT-2 model. Due to GPU memory constraints, we use a batch size of 20 during training and accumulate gradients for 3 steps, which results in an actual batch size of 60. Other implementation details are mostly adopted from Section 5.1.

In WikiBio, we followed a different strategy to add missing slots. Unlike E2E, WikiBio contains longer input tables and not all input information is present in the references, i.e., the first sentence of the Wiki article. Therefore, our search algorithm inserts a subset of input slots, determined by co-occurrence statistics on the validation dataset. As a heuristic, we select slots which occur at least in 10% tables of the dataset and are present in at least 10% output references.

Evaluation metrics. We included BLEU for reference, as it is the metric in Chen et al. (2020b). However, we still consider the PARENT scores as our main metric in our study, due to the evidence Dhingra et al. (2019).

We did not compute semantic coverage ratio for WikiBio, since it is known that the reference (the first sentence of a Wiki article) would not cover all the slots in the infobox. However, PARENT is able to measure the coverage of input slots as it considers both input and reference when computing precision, recall, and the F-score.

Results. Table 2 shows the result on WikiBio dataset. Since Chen et al. (2020b) do not report PARENT metrics for their fine-tuned GPT-2 model, we replicated the model by using their released code.[4] As seen from Lines 1–2, we achieved a sim-

[2] It should be mentioned that PPL may refer to very different evaluation protocols. In Chen et al. (2020a), for example, they use their trained model to evaluate the human-written references' PPL. Such protocol, although giving small PPL values, does not directly evaluate the generated text, and therefore, is not adopted in our study. By contrast, we used a third-party pre-trained language model, namely, GPT-2, to evaluate the PPL of our generated text. Different from Li et al. (2020), we did not fine-tune GPT-2 on our corpus. Our PPL approximately evaluates how fluent the generated text is as general English.

[3] https://github.com/DavidGrangier/wikipedia-biography-dataset

[4] We have not fully replicated Gong et al. (2020) due to the lack of their code, but nevertheless include BLEU for reference. This shows the superiority of our approach. Detailed comparison was not required by ACL guideline https://2021.aclweb.org/calls/papers/#citation-and-comparison

#	Model	#Train	BLEU	PARENT (P/R/F1)	PPL	AvgLen
1	GPT2+copy (Chen et al., 2020b)	p:100	29.5	–	–	–
2	GPT2+copy (our replication)	p:100	29.05	59.03 / 26.63 / 33.59	314.03	20.01
3	TableGPT2 (Gong et al., 2020)	p:100	34.5	–	–	–
4	T5	p:100	35.87	**65.21** / 29.59 / 38.00	219.03	17.35
5	T5 w/ self-train (Recomb)	p:100	**36.00**	64.74 / 29.58 / 37.91	219.40	17.27
6	T5 w/ S&L (Recomb)	p:100	35.41	64.10 / **30.23** / **38.34**	**218.48**	18.75
7	T5 w/ self-train	p:100, u:400	35.62	64.68 / 29.92 / 38.19	216.19	18.17
8	T5 w/ S&L	p:100, u:400	**35.92**	64.56 / **32.28** / **40.27**	**211.35**	19.84

Table 2: Test results on WikiBio (in the Humans domain). "p:" and "u:" denote the number of parallel and unlabeled training samples, respectively. "Recomb" means that we synthesize 400 samples by recombining table slots in the parallel corpus. "p:" and "u:" denote the number of parallel and unlabeled training samples, respectively.

ilar BLEU score to Chen et al. (2020b), showing that our replication is fair.

We apply our S&L to the WikiBio dataset. We see that, with 400 unlabeled tables, we improve the T5 model by 2–3 points in PARENT Recall and F1 (Lines 4, 7–8). This suggests that our model not only generates high-quality sentences in general for the data-to-text task, but also has a higher coverage of input slots due to the nature of PARENT metrics. We also obtain fluent sentences for this dataset, shown by relatively low PPL.[5] These results are generally consistent with E2E experiments.

Compared with previous state-of-the-art few-shot learning (Chen et al., 2020b), our setting uses extra 400 tables. To make a fair comparison, we synthesized 400 tables by recombining the slots without using any additional data. Comparing Line 6 and Line 2 suggests that, even without an unlabeled corpus, our approach still outperforms the previous state-of-the-art model in all metrics.

We also admit that recombining table slot does not give as good performance as using additional unlabeled tables (Line 6 vs. Line 8). A plausible reason is that new tables are able to train T5 with more slot values, and this is especially useful for few-shot data-to-text generation, where we only have a few hundred parallel samples. Recombining table slots cannot serve for this goal. Future research can be addressed here on effective data augmentation for data-to-text generation.

5.3 Analysis

In this part, we provide detailed analysis of our approach. Due to the limitation of space and available resources, we chose E2E as our testbed.

Human evaluation. We would like to conduct human evaluation for our model, as automatic metrics may not fully reflect the performance of a text generator. We selected a random subset of 50 samples and obtained the outputs from T5 self-train and T5 S&L. While the subset appears to be small, we will compute statistical significance to demonstrate that it suffices to draw a conclusion.

We asked three annotators to evaluate each table–text pair on three criteria: *coverage*, *fluency*, and *overall quality*. Coverage measures the number of input table slots present in the text divided by total number of input slots. Fluency measures if the sentence is clear, natural, and grammatically correct (3: Fluent, natural and grammatically correct; 2: Mostly fluent, with minor errors; 1: Not fluent, multiple grammatical errors). The annotators were also asked to assign an overall quality to each sentence (3: good quality; 2: average quality; 1: poor quality). Our human annotation was conducted in a strict blind fashion, i.e., samples were randomly shuffled and the annotator did not know the model of a generated sentence.

Table 3 presents human evaluation results. We see that the human-annotated coverage ratio is similar to automatic counting in Table 1. Our S&L achieves near-perfect semantic coverage, whereas a fine-tuned T5 with self-training only achieves 81.66% coverage. In terms of fluency, S&L is slightly worse than T5 self-training. However, the difference is 1/3 standard deviation, which is relatively small compared with our improvements in other aspects. The overall quality of our approach is considerably higher than the competing method by more than two standard deviations, showing the effectiveness of our approach. The human annoation results are generally consistent with our automatic measures (except BLEU).

Search and learning vs. Search for inference. An interesting analysis of our approach is to see

[5]The PPL for WikiBio sentences is higher than E2E because the the WikiBio corpus is more complex. Especially, WikiBio sentences contain quite a few proper nouns, such as the name of a person.

Input table

Slot	Value
Name	The Phoenix
Eat type	Restaurant
Food	Indian
PriceRange	£ 20-25
Customer Rating	High
Area	Riverside
Near	Crowne plaza hotel
Family Friendly	No

Reference 1: the phoenix is a restaurant that also serves indian food priced between £20-25, located near crowne plaza hotel on the riverside. it's customer rating is high, and the establishment is kids friendly. (All slots are present)

Reference 2: the phoenix, located near crowne plaza hotel on the riverside, is a restaurant that also serves indian food. it is kids friendly and food is priced between £20-25. (Missing slot: customer rating)

T5 few-shot fine-tuned: the phoenix is a restaurant that serves indian food in the price range of £20-25. it is near crowne plaza hotel. it has a high customer rating. (Missing slots: riverside, family friendly)

T5 self-train: the phoenix is a restaurant that serves indian food in the price range of £20-25. it is near crowne plaza hotel. (Missing slots: high, riverside, family friendly)

T5 search for inference: the phoenix is a restaurant that serves indian food in the price range of £20-25. it is near crowne plaza high customer rating in riverside area not family friendly hotel. (All slots are present, but the sentence is not fluent)

T5 S&L: in riverside area the phoenix is a restaurant that serves indian food in the price range of £20-25. it is near crowne plaza hotel. it has a high customer rating and is not family-friendly. (All slots are present)

Figure 2: A case study of few-shot data-to-text generation on the E2E dataset.

Model	Coverage	Fluency	Overall Quality
T5 w/ self-train	81.66%	**2.88±0.32**	2.1±0.34
T5 w/ S&L	**99.58%**	2.75±0.43	**2.81±0.39**
p-value	6.08e-24	0.00664	3.84e-22

Table 3: Human evaluation results on the E2E dataset. The p-values are given by two-sided Wilcoxon paired test. It only shows whether our annotated subset has collected enough evidence for drawing a conclusion, instead of how different two models are. We show the standard deviation, which roughly estimates if the gap is relatively large or not.

Model	PARENT (P/R/F1)	Inf. Time	PPL
Search for inference	**65.71 / 60.17/ 61.67**	**113.4** secs	**234.19**
S&L	66.97 / 63.63 / 64.29	78.05 secs	160.40

Table 4: Search and learning vs. search for inference. Inference time was obtained by predicting the test set on a single V100 GPU.

how search and learning (S&L) improves the search itself. This can be seen by performing search for inference on the test set. From Table 4, we observe that S&L largely improves the search results in terms of all metrics. Especially, the PPL of S&L is considerably smaller than search for inference. This shows that the second-stage fine-tuning (Section 4.3) not only learns from the search results for higher semantic coverage, but also smooths out the search noise and yields better sentences in general.

In addition, S&L has a better inference efficiency. Despite our batch implementation and the V100 GPU device, search for inference takes 45% more time than S&L in inference. This shows that our approach is efficient in practice.

Case Study. We conduct a case study in Figure 2. There are 7 references for this data sample, and we present two references to illustrate that the input slots may be missing even in references. We see that T5 (fine-tuned with few-shot $\mathcal{D}_p$ or further self-trained with $\mathcal{D}_u$) yields fluent sentences, and does not generate wrong information as addressed in (Gong et al., 2020). However, a few input slots are missing in T5's output. If we perform search for inference, we are guaranteed to have perfect slot coverage, but the sentence may not be fluent, such as "*it is near crowne plaza high customer rating in riverside area not family friendly hotel*". Our S&L approach yields a fluent sentence with a high semantic coverage.

6 Conclusion and Future Work

In this work, we present a search-and-learning approach to address the low coverage problem for few-shot data-to-text generation. We first fine-tune the pre-trained T5 language model based on a small parallel corpus. Then, we use the T5 to predict on an unlabeled corpus, and search for higher semantic coverage. The T5 is further fine-tuned with search results. Experiments on E2E and WikiBio datasets show that our model achieves high performance than previous approaches to few-shot data-to-text generation, largely closing the gap between few-shot and fully supervised learning

In future work, we would apply our S&L framework to fidelity-oriented text generation by revising counterfactual text and further learning from the edit results. This would benefit various NLG tasks, such as paraphrasing and summarization.

Acknowledgements

Shailza Jolly was supported by the TU Kaiserslautern CS Ph.D. scholarship program, the BMBF project XAINES (Grant 01IW20005), and the NVIDIA AI Lab (NVAIL) program.

References

Wenhu Chen, Jianshu Chen, Yu Su, Zhiyu Chen, and William Yang Wang. 2020a. Logical natural language generation from open-domain tables. In *ACL*, pages 7929–7942.

Zhiyu Chen, Harini Eavani, Wenhu Chen, Yinyin Liu, and William Yang Wang. 2020b. Few-shot NLG with pre-trained language model. In *ACL*, pages 183–190.

Robert Dale and Ehud Reiter. 1997. Building applied natural language generation systems. *Journal of Natural Language Engineering*, 3(1):57–87.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186.

Bhuwan Dhingra, Manaal Faruqui, Ankur Parikh, Ming-Wei Chang, Dipanjan Das, and William Cohen. 2019. Handling divergent reference texts when evaluating table-to-text generation. In *ACL*, pages 4884–4895.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *HLT*, pages 138–145.

Yue Dong, Zichao Li, Mehdi Rezagholizadeh, and Jackie Chi Kit Cheung. 2019. EditNTS: An neural programmer-interpreter model for sentence simplification through explicit editing. In *ACL*, pages 3393–3402.

Heng Gong, Yawei Sun, Xiaocheng Feng, Bing Qin, Wei Bi, Xiaojiang Liu, and Ting Liu. 2020. TableGPT: Few-shot table-to-text generation with table structure reconstruction and content matching. In *COLING*, pages 1978–1988.

Juraj Juraska, Panagiotis Karagiannis, Kevin Bowden, and Marilyn Walker. 2018. A deep ensemble model with slot alignment for sequence-to-sequence natural language generation. In *NAACL*, pages 152–162.

R. Koncel-Kedziorski, Hannaneh Hajishirzi, and Ali Farhadi. 2014. Multi-resolution language grounding with weak supervision. In *EMNLP*, pages 386–396.

Karen Kukich. 1983. Design of a knowledge-based report generator. In *ACL*, pages 145–150.

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *ACL-COLING*, pages 704–710.

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.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *EMNLP*, pages 1203–1213.

Jingjing Li, Zichao Li, Lili Mou, Xin Jiang, Michael R Lyu, and Irwin King. 2020. Unsupervised text generation by learning from search. In *NeurIPS*.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: A simple approach to sentiment and style transfer. In *NAACL-HLT*, pages 1865–1874.

Percy Liang, Michael Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *ACL-IJCNLP*, pages 91–99.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81.

Tianyu Liu, Fuli Luo, Qiaolin Xia, Shuming Ma, Baobao Chang, and Zhifang Sui. 2019. Hierarchical encoder with auxiliary supervision for neural table-to-text generation: Learning better representation for tables. In *AAAI*, pages 6786–6793.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2018. Table-to-text generation by structure-aware seq2seq learning. In *AAAI*, pages 4881–4888.

Xianggen Liu, Lili Mou, Fandong Meng, Hao Zhou, Jie Zhou, and Sen Song. 2020. Unsupervised paraphrasing by simulated annealing. In *ACL*, pages 302–312.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *ICLR*.

Kathleen McKeown. 1992. *Text Generation*. Cambridge University Press.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The E2E dataset: New challenges for end-to-end generation. In *SIGDIAL*, pages 201–206.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*.

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. *JMLR*, 21:1–67.

Verena Rieser and Oliver Lemon. 2009. Natural language generation as planning under uncertainty for spoken dialogue systems. In *EACL*, pages 683–691.

Raphael Schumann, Lili Mou, Yao Lu, Olga Vechtomova, and Katja Markert. 2020. Discrete optimization for unsupervised sentence summarization with word-level extraction. In *ACL*, pages 5032–5042.

Sheng Shen, Daniel Fried, Jacob Andreas, and Dan Klein. 2019. Pragmatically informative text generation. In *NAACL-HLT*, pages 4060–4067.

Amanda Stent, Rashmi Prasad, and Marilyn Walker. 2004. Trainable sentence planning for complex information presentations in spoken dialog systems. In *ACL*, pages 79–86.

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 *NIPS*, pages 5998–6008.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. CIDEr: Consensus-based image description evaluation. In *CVPR*, pages 4566–4575.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2018. Learning neural templates for text generation. In *EMNLP*, pages 3174–3187.

Xiaojin Zhu and Andrew B Goldberg. 2009. Introduction to semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 3(1):1–130.

A Complete List of Rules

As explained in Section 4.2, we insert a template-based phrase if a slot value does not appear in the output text. Below is the complete list of the rules we used in our experiments. SN refers to the slot name, and SV refers to the slot value. Despite the simplicity of the rules, our search-and-learning (S&L) approach improves the semantic coverage to a large extent for few-shot data-to-text generation.

If	**Then the phrase template is**
SN = food	SV food
SN = pricerange; SV is a number	price range SV
SN = pricerange; SV is a string	SV price range
SN = eattype	SV
SN = name	SV
SN = near	near SV
SN = family friendly; SV is yes	family friendly
SN = family friendly; SV is no/not	not family friendly
SN = customer rating	SV customer rating
SN = area	in SV area

Table 5: Rules for the E2E dataset.

If	**Then the phrase template is**
SN = fullname	SV
SN = birth date	born on SV
SN = currentclub	plays for SV
SN = nationality	SV
SN = position	SV
SN = occupation	is a SV
SN = death rate	died on SV
SN = party	serving in SV party
SN = birth place	born in SV

Table 6: Rules for the WikiBio dataset.