PRELIMINARY PREPRINT VERSION: DO NOT CITE
The AAAI Digital Library will contain the published
version some time after the conference.

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

Shailza Jolly[1,2], **Zi Xuan Zhang**[3], **Andreas Dengel**[1,2], **Lili Mou**[3]

[1] TU Kaiserslautern, Germany   [2] DFKI GmbH, Germany
[3] Dept. Computing Science, Alberta Machine Intelligence Institute (Amii), University of Alberta, Canada
shailza.jolly@dfki.de,   zixuan7@ualberta.ca,
andreas.dengel@dfki.de,   doublepower.mou@gmail.com

Abstract

Data-to-text generation systems aim to generate text descriptions based on input data (often represented in the tabular form). A typical system uses huge training samples for learning the correspondence between tables and texts. However, large training sets are expensive to obtain, limiting the applicability of these approaches in real-world scenarios. In this work, we focus on few-shot data-to-text generation. We observe that, while fine-tuned pretrained language models may generate plausible sentences, they suffer from the *low semantic coverage* problem in the few-shot setting. In other words, important input slots tend to be missing in the generated text. To this end, we propose a search-and-learning approach that leverages pretrained language models but inserts the missing slots to improve the semantic coverage. We further fine-tune our system based on the search results to smooth out the search noise, yielding better-quality text and improving inference efficiency to a large extent. Experiments show that our model achieves high performance on E2E and WikiBio datasets. Especially, we cover 98.35% of input slots on E2E, largely alleviating the low coverage problem.[1]

Introduction

Data-to-text generation is a task that converts structured data 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, Dušek, and Rieser 2017), biographies (Lebret, Grangier, and Auli 2016), and weather forecasts (Liang, Jordan, and Klein 2009).

Traditional approaches to natural language generation (NLG) use handcrafted rules with statistics (Langkilde and Knight 1998; Stent, Prasad, and Walker 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, Grangier, and Auli 2016; Liu et al. 2018). These models use massive parallel training data, for example, 42K table–text training pairs

in the E2E dataset (Novikova, Dušek, and Rieser 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) apply few-shot learning to data-to-text generation by fine-tuning pre-trained language models (LMs) with a copy mechanism. Pretrained LMs learn generic knowledge of natural language from massive unlabeled corpora, and thus are able to generate plausible text with fewer samples than traditional neural networks. However, we observe that fine-tuned LMs fail to fully learn the correspondence between input and output in the few-shot setting. 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 pre-trained T5 language model (Raffel et al. 2020), similar to previous work of Chen et al. (2020b). To address the low coverage problem, we iteratively insert a missing slot into the generated sentence. We try all possible positions for the insertion, 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 semantic coverage of the input slots. Also, our model is efficient for generating sentences and does not increase the inference complexity; it also yields more fluent sentences compared with search-based text generation (Liu et al. 2020).

In summary, our main contributions include: 1) We address the *low semantic coverage* problem in few-shot data-to-text generation. 2) To the best of our knowledge, we are the first to propose search-and-learning approaches for data-to-text generation, where we incorporate traditional template-based methods as search actions to insert missing slots. 3) We conducted 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. Especially, we achieve 98.35% coverage on the E2E dataset.
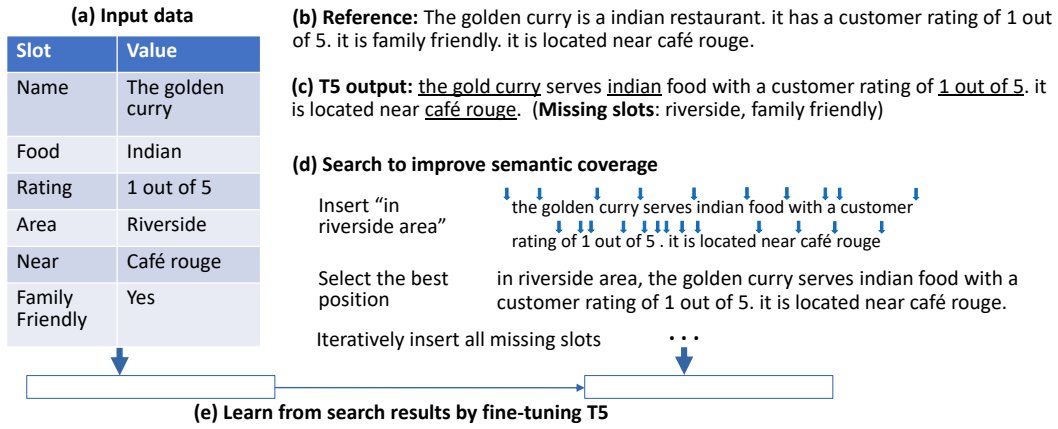
[1]Our code and output are available at https://github.com/shailzajolly/FSDT

(a) Input data

Slot	Value
Name	The golden curry
Food	Indian
Rating	1 out of 5
Area	Riverside
Near	Café rouge
Family Friendly	Yes

(b) Reference: The golden curry is a indian restaurant. it has a customer rating of 1 out of 5. it is family friendly. it is located near café rouge.

(c) T5 output: the gold curry serves indian food with a customer rating of 1 out of 5. it is located near café rouge. (**Missing slots**: riverside, family friendly)

(d) Search to improve semantic coverage

Insert "in riverside area" the golden curry serves indian food with a customer rating of 1 out of 5 . it is located near café rouge

Select the best position in riverside area, the golden curry serves indian food with a customer rating of 1 out of 5. it is located near café rouge.

Iteratively insert all missing slots ⋯

(e) Learn from search results by fine-tuning T5

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

Related Work

Data-to-Text Generation. Generating human-readable 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, Jordan, and Klein 2009; Koncel-Kedziorski, Hajishirzi, and Farhadi 2014). However, the generated text usually lacks flexibility and diversity.

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

Recently, Chen et al. (2020b) address few-shot learning for data-to-text generation, where they assume a small parallel corpus is available. They propose to fine-tune pre-trained language models (LMs) with a copy mechanism. We observe that, although such fine-tuned LMs generate fluent sentences, they suffer from the problem of low semantic coverage, as it is difficult to "force" a copy mechanism to copy the entire source information.

In fact, Dhingra et al. (2019) point 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 generation models, 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.

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; Kumar et al. 2020).

Their 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 model performance and inference efficiency.

Our paper adopts the search-and-learning framework in Li et al. (2020), but differs from previous 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 word edits in a supervised way (Dong et al. 2019), or treat rule-edited text as input (Li et al. 2018; Wang et al. 2019). We instead perform editing as search steps and further learn from the results of editing.

Problem Formulation

Data-to-text generation aims to generate a natural language description for structured input data; we consider a common setting, where the input is tabular data. For each 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 $\mathbf{y} = (y_1, y_2, \cdots, y_n)$ that describes the given input $\mathbf{T}$. Notice that the table $\mathbf{T}$ is different for each sample, but we may omit the sample index for clarity.

In this work, we consider the few-shot setting, where we have a small parallel corpus $\mathcal{D}_p = \{(\mathbf{T}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{M}$ and another small unlabeled corpus $\mathcal{D}_u = \{\mathbf{T}_u^{(i)}\}_{i=1}^{N}$, where $\mathbf{T}_u^{(i)}$ is a different table than $\mathbf{T}^{(i)}$. 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 labor and also helps to alleviate the cold-start problem of new NLG tasks. 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.

Proposed Model

In our approach, we first fine-tune a pre-trained language model (LM) for conditional text generation based on input tables. The large model capacities and extensive pre-training are among the reasons why LMs would help us with generating fluent sentences in the few-shot setting.

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. 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.

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 (Vaswani et al. 2017), pre-trained on multiple NLP tasks and achieves state-of-the-art performance on question answering, document summarization, sentiment classification, etc.

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 linearize 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 a corpus with 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

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

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 1, 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. This can be done by either a verbatim match of the slot value or a soft match based on script that uses regular expressions to find missing slots (Dušek, Howcroft, and Rieser 2019). While verbatim match may be strict and noisy, our results will show that it achieves equally good performance in our S&L approach.

If a slot value v_i does not appear in the output, we insert a phrase $\tilde{v}_i$ that contains the original slot value v_i along with possible supporting prepositions. 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 as either "family friendly" or "not family friendly". Designing these phrases does not require much human labor, as we have no more than 10 phrases for each dataset, and quite a few of them are simply copying the slot value. The complete list of our phrases is shown in Appendix.

For every missing slot, we determine the most appropriate position for inserting the slot. 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 Eq. (1), shown in Figure 1d.

This process is repeated in a greedy fashion for all missing slots that we would like to insert.[2] Our approach can be thought of as an optimization towards

$$\text{maximize } P(\mathbf{y}|\mathbf{T}) \qquad (3)$$
$$\text{subject to } v_i \in \mathbf{y}, \forall i$$

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 greedily.

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 a heuristically defined objective.

Our approach is also related to template-based text generation systems in the early years (Langkilde and Knight

---

[2]For our E2E experiment, we would insert all slots. If a task does not require that the output sentence contains all slots, we may select the desired slots by statistics (see the WikiBio experiment).

Algorithm 1: Search and Learn

Input: Small parallel data $\mathcal{D}_p = \{(\mathbf{T}^{(m)}, \mathbf{y}^{(m)})\}_{m=1}^M$
Small unlabeled data $\mathcal{D}_u = \{\mathbf{T}^{(n)}\}_{n=1}^N$
Pre-trained language model T5
Output: Few-shot learned data-to-text model
▷ First-stage fine-tuning T5
for $(\mathbf{T}, \mathbf{y}) \in \mathcal{D}_p$ *in each epoch* **do**
   Fine-tune T5 by minimizing $-\log P(\mathbf{y}|\mathbf{T})$
▷ Search to improve semantic coverage
$\widetilde{\mathcal{D}}_p = \emptyset$
for $\mathbf{T}_u \in \mathcal{D}_u$ **do**
   $\hat{\mathbf{y}}_{\text{search}} = \text{T5}(\mathbf{T}_u)$ ▷ search to be performed
   for *missing slot* $(s, v) \in \mathbf{T}_u$ *that* $v \notin \hat{\mathbf{y}}_{search}$ **do**
      Update $\hat{\mathbf{y}}_{\text{search}}$ by inserting v with templates into the most appropriate position
   $\widetilde{\mathcal{D}}_p = \widetilde{\mathcal{D}}_p \cup \{(\mathbf{T}_u, \hat{\mathbf{y}}_{\text{search}})\}$
▷ Second-stage fine-tuning T5
for $(\mathbf{T}, \mathbf{y}) \in \mathcal{D}_p \cup \widetilde{\mathcal{D}}_p$ *in each epoch* **do**
   Fine-tune T5 by minimizing $-\log P(\mathbf{y}|\mathbf{T})$
Return: Two-stage fine-tuned T5

1998; Stent, Prasad, and Walker 2004). Our work differs significantly, as we use rules only for revision, rather than for generation. Traditional rule-based systems often generate inflexible and disfluent text. We will have a learning component that learns from the search results to smooth out disfluent text.

Second-Stage Fine-Tuning T5 with Search Results

The search approach in our previous section ensures a high semantic coverage of the output sentence, but 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 when evaluating multiple candidate outputs given a data sample. To address them, 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 a small unlabeled corpus $\mathcal{D}_u$ containing input tables only. In practice, $\mathcal{D}_u$ can be either obtained inexpensively or 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 and perform search for higher semantic coverage. The search result is treated as a pseudo-groundtruth, denoted by $\hat{\mathbf{y}}_{\text{search}}^{(i)}$. This in turn yields a pseudo-parallel corpus $\widetilde{\mathcal{D}}_p = \{(\mathbf{T}_u^{(i)}, \hat{\mathbf{y}}_{\text{search}}^{(i)}) : \mathbf{T}_u^{(i)} \in \mathcal{D}_u\}$. It is mixed with the original parallel corpus for further fine-tuning T5. In other words, our dataset becomes $\mathcal{D}_p \cup \widetilde{\mathcal{D}}_p$, and T5 is further fine-tuned by the same cross-entropy loss as Eq. (2). Algorithm 1 summarizes our training algorithm.

Inference

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

In this way, our inference efficiency is improved compared with 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 , T5 this time is explicitly trained with pseudo-groundtruth that has high semantic coverage. Experiments will show our S&L approach achieves near-perfect semantic coverage on the E2E dataset.

Experiments

Experiment I: E2E Dataset

Dataset. In this experiment, we used the E2E dataset[3] (Novikova, Dušek, and Rieser 2017), which is a crowd-sourced 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 the model 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, Dušek, and Rieser 2017), including BLEU (Papineni et al. 2002), NIST (Doddington 2002), METEOR (Lavie and Agarwal 2007), ROUGE-L (Lin 2004), and CIDEr (Vedantam, Lawrence Zitnick, and Parikh 2015).

Recently, Dhingra et al. (2019) observe that BLEU does not correlate well to human satisfaction for data-to-text generation. 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 consider PARENT as the main metric. Specifically, we used the word-overlap version PARETNT-W in our paper.

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 (Hard Coverage), 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 consider slot error rate (SER, Dušek, Howcroft, and Rieser 2019), designed specifically for the E2E dataset. The metric checks if all E2E slot values are present, or missing/incorrect[4] in the output text based on manually designed regular expressions. Unlike verbatim matching, SER accounts for soft matching, and we consider $1 - \text{SER}$ as Soft Coverage. We also conducted a human evaluation on a randomly selected subset of test samples, and the coverage percentage (Table 3) is close to these automatic metric.

---

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

[4]SER breakdown is presented in Appendix.

#	Model	#Train	BLEU	NIST	METEOR	RougeL	CIDEr	PARENT (P/R/F1)	PPL	AvgLen	Hard Coverage	SER	Soft Coverage
1	TGEN	p:42K	65.93	8.61	44.83	68.50	2.23	–	–	–	–	4.27%	95.73%
2	SLUG	p:42K	66.19	8.61	44.54	67.72	–	–	–	–	–	–	–
3	S_1^R (Shen et al. 2019)	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	97.50%	2.62%	97.38%
5	T5	p:2100	62.45	8.30	44.10	67.15	2.17	64.25 / 61.69 / 62.00	136.54	24.82	96.21%	3.72%	96.28%
6	T5	p:420	**61.72**	7.96	40.52	65.61	1.96	65.63 / 57.25 / 60.10	**141.61**	21.97	84.19%	16.68%	84.32%
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	81.82%	17.97%	82.03%
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	98.35%	1.84%	98.16%
9	T5 S&L w/ SER	p:420, u:1680	60.89	**8.14**	**43.71**	**66.76**	2.07	65.16 / 62.97 / 63.04	170.26	25.71	**99.46%**	**0.80%**	**99.20%**

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. 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. Hard Coverage measures the verbatim coverage of slot values presented in text. The slot error rate (SER, Dušek, Howcroft, and Rieser 2019) measures the percentage of missing, added, or wrong slots in the generated sentence. We consider Soft Coverage as $1 - \text{SER}$.

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, Dušek, and Rieser 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, both hard and soft coverage scores drop quickly from more than 97% to around 84.19%.

We would like to see if a small unlabeled dataset $\mathcal{D}_u$, which contains tables only, could help the performance. We experimented with 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, we observe from Lines 6 and 7 that such strategy does not help the performance much.

Finally, we applied two variants of our S&L approach, where Line 8 is a variant that determines missing slots by verbatim match, and Line 9 determines missing slots by SER. Results show that both variants achieve higher performance than other few-shot models in terms of most metrics. Especially, our model achieve 98–99% coverage of input slots, mostly solving the low coverage problem.

Based on the numerical results, it appears that our model generate less fluent sentences, given by high perplexity (PPL) scores.[5] 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 (Lines 8 and 9) 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 and coverage scores to the fully supervised setting (Line 4).

Experiment II: WikiBio Dataset

Dataset. We further evaluate our approach on the Humans domain of WikiBio data[6] (Lebret, Grangier, and Auli 2016). It 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), 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

---

[5]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.

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

#	Model	#Train	BLEU	PARENT (P/R/F1)	PPL	AvgLen	Coverage (Table)	Coverage (Reference)
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	27.07%	55.27%
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	38.45%	76.61%
5	T5 self-train (Recomb)	p:100	**36.00**	64.74 / 29.58 / 37.91	219.40	17.27	38.20%	76.01%
6	T5 S&L (Recomb)	p:100	35.41	64.10 / **30.23** / **38.34**	**218.48**	18.75	**41.29%**	**76.75%**
7	T5 self-train	p:100, u:400	35.62	64.68 / 29.92 / 38.19	216.19	18.17	40.22%	76.85%
8	T5 S&L	p:100, u:400	**35.92**	64.56 / **32.28** / **40.27**	**211.35**	19.84	**42.45%**	**79.14%**
9	T5 S&L (cosine similarity)	p:100, u:400	35.44	63.63 / 31.32 / 39.36	233.18	18.92	41.28%	75.68%

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. The bold font indicates the best performance in each group that also outperforms the baselines in Lines 1–4. Coverage scores are computed against the input table and the reference text, respectively.

size of 60. Other implementation details are mostly adopted from Experiment I.

In WikiBio, we used a different strategy to add missing slots. Unlike E2E, WikiBio contains longer input tables and not all input slots are present in the references (the first sentence of the Wiki article). Therefore, our search algorithm inserts a subset of input slots, determined by co-occurrence statistics on the few-shot training 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-W scores as the main metric in our study, due to the evidence from Dhingra et al. (2019).

For semantic coverage, we mainly consider the hard version, because no soft coverage has been developed for WikiBio and because our E2E experiments show that hard and soft coverages are generally close to each other. It is noted that WikiBio does not aim to cover every input slot; thus, we also compute the coverage against the reference.

Results. Table 2 shows the results on WikiBio. Since Chen et al. (2020b) did not report PARENT metrics for their fine-tuned GPT-2 model, we replicated the model by using their released code.[7] As seen from Lines 1–2, we achieved a similar BLEU score to Chen et al. (2020b), showing that our replication was fair.

We applied our S&L approach to the WikiBio dataset. We see that, with 400 unlabeled tables, we improve the T5 model by 2–3 points in terms of 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. This is further confirmed by our coverage scores. Relatively low PPL shows that our sentences are fluent.[8]

We also implemented a variant that determines missing slots by thresholding cosine similarity of embeddings

(Line 9). Different from SER which is specifically engineered for E2E, the cosine similarity here is generic and does not work well compared with our verbatim matching. This further confirms that our approach is simple yet effective in alleviating the low semantic coverage problem.

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

We observe that recombining table slot does not give as good performance as using additional unlabeled tables (Lines 6 vs. 8). A plausible reason is that new tables are able to train T5 with more slot values, which 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.

Analysis

In this part, we provide detailed analysis of our approach. Due to the limit of space and resources, we chose E2E and the standard variant (Line 8, Table 1) as our testbed.

Human evaluation. We conducted 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 may appear to be small, we computed 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 the 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; 2: average; 1: poor). Our human annotation was conducted in a strict blind fashion, i.e., samples were randomly shuffled and the annotator

---

[7]Gong et al. (2020) did not release code or output; thus some metric evaluations are unavailable. Nevertheless, the BLEU score shows the superiority of our approach.

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

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

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

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

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

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

T5 S&L: in <u>riverside</u> area <u>the phoenix</u> is a <u>restaurant</u> that serves <u>indian</u> food in the price range of <u>£20-25</u>. it is near <u>crowne plaza hotel</u>. it has a <u>high</u> customer rating and is <u>not family-friendly</u>. (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 E2E. 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 or not, 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)	InfTime	RelTime	PPL
S&L	**66.97/63.63/64.29**	**78.05**	**1x**	**160.40**
Search for inf.	65.71/60.17/61.67	113.4	1.45x	234.19

Table 4: Search and learning vs. search for inference. <u>Inf</u>erence time (in seconds) and <u>Rel</u>ative time were obtained by predicting the test set on a single V100 GPU.

did not know the model of a generated sentence.

Table 3 presents human evaluation results. We observe that the human-annotated coverage ratio is similar to automatic counting in Table 1. Our S&L achieved near-perfect semantic coverage, whereas a fine-tuned T5 with self-training only achieves 81.66% coverage. In both models, annotators did not observe false information.

In terms of fluency, S&L behaves slightly worse than T5 self-training. However, the difference is one-third of a 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 annotation results are generally consistent with our automatic measures.

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

Appendices. We show the complete list of rules and slot error rate (SER) details in appendices, available at https://arxiv.org/abs/2112.02770.

Conclusion

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 higher performance than previous approaches to few-shot data-to-text generation, largely closing the gap between few-shot and fully supervised learning.

Acknowledgments

References

Chen, W.; Chen, J.; Su, Y.; Chen, Z.; and Wang, W. Y. 2020a. Logical natural language generation from open-domain tables. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 7929–7942.

Chen, Z.; Eavani, H.; Chen, W.; Liu, Y.; and Wang, W. Y. 2020b. Few-shot NLG with pre-trained language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 183–190.

Dale, R.; and Reiter, E. 1997. Building applied natural language generation systems. *Natural Language Engineering*, 3(1): 57–87.

Dhingra, B.; Faruqui, M.; Parikh, A.; Chang, M.-W.; Das, D.; and Cohen, W. 2019. Handling divergent reference texts when evaluating table-to-text generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 4884–4895.

Doddington, G. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the Second International Conference on Human Language Technology Research*, 138–145.

Dong, Y.; Li, Z.; Rezagholizadeh, M.; and Cheung, J. C. K. 2019. EditNTS: An neural programmer-interpreter model for sentence simplification through explicit editing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 3393–3402.

Dušek, O.; Howcroft, D. M.; and Rieser, V. 2019. Semantic noise matters for neural natural language generation. In *Proceedings of the 12th International Conference on Natural Language Generation*, 421–426.

Gong, H.; Sun, Y.; Feng, X.; Qin, B.; Bi, W.; Liu, X.; and Liu, T. 2020. TableGPT: Few-shot table-to-text generation with table structure reconstruction and content matching. In *Proceedings of the 28th International Conference on Computational Linguistics*, 1978–1988.

Juraska, J.; Karagiannis, P.; Bowden, K.; and Walker, M. 2018. A deep ensemble model with slot alignment for sequence-to-sequence natural language generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 152–162.

Koncel-Kedziorski, R.; Hajishirzi, H.; and Farhadi, A. 2014. Multi-resolution language grounding with weak supervision. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 386–396.

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

Kumar, D.; Mou, L.; Golab, L.; and Vechtomova, O. 2020. Iterative edit-based unsupervised sentence simplification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 7918–7928.

Langkilde, I.; and Knight, K. 1998. Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, 704–710.

Lavie, A.; and Agarwal, A. 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*, 228–231.

Lebret, R.; Grangier, D.; and Auli, M. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 1203–1213.

Li, J.; Jia, R.; He, H.; and Liang, P. 2018. Delete, retrieve, generate: A simple approach to sentiment and style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 1865–1874.

Li, J.; Li, Z.; Mou, L.; Jiang, X.; Lyu, M. R.; and King, I. 2020. Unsupervised text generation by learning from search. In *Advances in Neural Information Processing Systems*.

Liang, P.; Jordan, M.; and Klein, D. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, 91–99.

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

Liu, T.; Luo, F.; Xia, Q.; Ma, S.; Chang, B.; and Sui, Z. 2019. Hierarchical encoder with auxiliary supervision for neural table-to-text generation: Learning better representation for tables. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 6786–6793.

Liu, T.; Wang, K.; Sha, L.; Chang, B.; and Sui, Z. 2018. Table-to-text generation by structure-aware seq2seq learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 4881–4888.

Liu, X.; Mou, L.; Meng, F.; Zhou, H.; Zhou, J.; and Song, S. 2020. Unsupervised paraphrasing by simulated annealing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 302–312.

Loshchilov, I.; and Hutter, F. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

McKeown, K. 1992. *Text Generation*. Cambridge University Press.

Novikova, J.; Dušek, O.; and Rieser, V. 2017. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, 201–206.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 311–318.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21: 1–67.

Rieser, V.; and Lemon, O. 2009. Natural language generation as planning under uncertainty for spoken dialogue systems. In *Proceedings of the 12th Conference of the European Chapter of the ACL*, 683–691.

Schumann, R.; Mou, L.; Lu, Y.; Vechtomova, O.; and Markert, K. 2020. Discrete optimization for unsupervised sentence summarization with word-level extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 5032–5042.

Sha, L.; Mou, L.; Liu, T.; Poupart, P.; Li, S.; Chang, B.; and Sui, Z. 2018. Order-planning neural text generation from structured data. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 5414–5421.

Shen, S.; Fried, D.; Andreas, J.; and Klein, D. 2019. Pragmatically informative text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4060–4067.

Stent, A.; Prasad, R.; and Walker, M. 2004. Trainable sentence planning for complex information presentations in spoken dialog systems. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, 79–86.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 5998–6008.

Vedantam, R.; Lawrence Zitnick, C.; and Parikh, D. 2015. CIDEr: Consensus-based image description evaluation. In *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition*, 4566–4575.

Wang, Y.; Wu, Y.; Mou, L.; Li, Z.; and Chao, W. 2019. Harnessing pre-trained neural networks with rules for formality style transfer. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, 3573–3578.

Wiseman, S.; Shieber, S.; and Rush, A. 2018. Learning neural templates for text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 3174–3187.

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