

Prefix-Tuning: Optimizing Continuous Prompts for Generation Tasks

First Author

Affiliation / Address line 1
Affiliation / Address line 2
Affiliation / Address line 3
email@domain

Second Author

Affiliation / Address line 1
Affiliation / Address line 2
Affiliation / Address line 3
email@domain

Abstract

Fine-tuning is the de facto way of leveraging large pretrained language models for downstream tasks. However, it modifies all the language model parameters and therefore necessitates storing all of them for each task. In this paper, we propose prefix-tuning, a lightweight alternative to fine-tuning for natural language generation tasks, which keeps LM parameters frozen, but optimizes a small *continuous task-specific prompt*. We apply prefix-tuning to GPT-2 for table-to-text generation and to BART for summarization. We find that with only 0.1% of the parameters, prefix-tuning obtains comparable performance in the full data setting, outperforms fine-tuning in low-data settings, and extrapolates better to topics unseen during training.

1 Introduction

Fine-tuning is the prevalent paradigm for using large pretrained language models (Radford et al., 2019; Devlin et al., 2019) in downstream tasks (e.g., summarization), but it requires updating and storing all the parameters. Consequently, **to build and deploy NLP systems that rely on large pretrained language models, one currently needs to store a modified copy of the language model parameters for each task.** This can be prohibitively expensive, given the size of current language models, for example, GPT-2 has 774M parameters and GPT-3 has 175B parameters (Brown et al., 2020; Radford et al., 2019). Also, fine-tuning is prone to overfitting and suffers instability in low-data settings (Kornblith et al., 2019; Dodge et al., 2020).

In this paper, we propose *prefix-tuning*, which is a lightweight yet effective alternative to fine-tuning. We focus on the task of natural language generation (NLG), but our method can be easily extended to other NLP tasks. Consider a task of generating textual description of a data table, as shown in

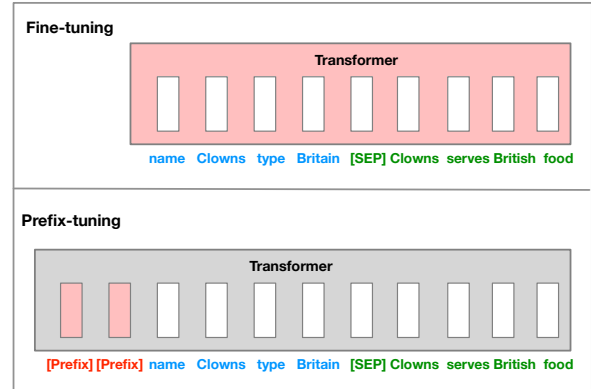


Figure 1: Our instantiation of the prefix-tuning (lower box), compared with fine-tuning (upper box) for table-to-text generation. The prefix is prepended before the task input (in blue) and output (in green). Trainable modules are colored by red. Fine-tuning optimizes the full transformer parameters (the red transformer box), whereas prefix-tuning freeze them and only optimize the soft prefix (the red blocks). Note that each vertical block is a stack of activation layers at one time step.

Figure 1, where the task input is a linearized table (e.g. “name Clown type Britain”) and the output is a textual description (e.g. “Clowns serves British food”). Prefix-tuning prepends a *task-specific, continuous* prompt **before reading any task input**. The continuous prompt, also denoted as the *prefix* (i.e. the red activation blocks), is the stacks of activation layers on top of the fictitious tokens [prefix]. **Instead of using the transformer’s parameters to compute the prefix activations bottom up, they are directly trainable.** The prefix affects the activations of all tokens to its right, including the task input and the task output, and therefore capable of steering the language model distribution to specialize for a particular task. In contrast to finetuning, which updates all transformer parameters and therefore necessitates storing a tuned copy of the model for each task (e.g. 774M parameters), prefix-tuning only optimizes the prefix activations and the same learned prefix is used for all instances of a given

task. Therefore we only need one copy of the large language model and for each task we need to store the learned prefix, yielding a very small overhead for each additional task (e.g. 250K parameters for the table-to-text task). When these pretrained models are deployed on edge devices with limited storage, prefix-tuning is more advantageous as the number of tasks increases¹.

We discover that despite only storing 1000x fewer parameters than fine-tuning, prefix-tuning can maintain a comparable performance in the full data setting and outperforms fine-tuning in both low-data and extrapolation (i.e. generalization to topics that are unseen during training) settings.

Prefix-tuning is a form of continuous prompting. Natural language prompts are particularly effective in GPT-3 and its in-context learning framework. Prompting in GPT-3 refers to discrete natural language instructions (e.g. *TL;DR* for summarization) followed by a few examples prepended before generation. The learned prefix in prefix-tuning can be viewed as the continuous version of GPT-3 instructions. Despite GPT-3’s outstanding performance in few-shot settings, transformers can only condition on a bounded-length context (e.g., 2048 tokens for GPT-3). Therefore, in-context learning is unable to exploit large training sets that don’t fit into the context window.

Discrete prompt design has been explored in prior works for models like BERT and RoBERTa (Jiang et al., 2020; Schick and Schütze, 2020). Researchers engineered natural language prompts either manually or using models that were constrained to be fluent. However, fluent prompts may be suboptimal in terms of performance. Auto-Prompt (Shin et al., 2020) searches for a sequence of trigger words (without fluency constraints) and concatenates them with the input to steer BERT to improve classification accuracy. The past work only focuses on classification or knowledge extraction tasks. We focus on generation tasks, which are harder to steer with just a few words (§ 7.2) and discrete optimization is computationally difficult. In contrast, we optimize the continuous activation space of the prefix, which is sufficiently expressive to steer the generation for a targeted task (§6) and we can use gradient-based optimization to tune the prefix.

¹Let N be the number of LM parameter and n be the number of prefix parameters, with $N \gg n$. Fine-tuning k tasks stores $k \cdot N$ parameters, whereas prefix-tuning stores only $N + k \cdot n$

Our approach can be viewed from the perspective of lightweight fine-tuning, whose goal is to freeze most of the pretrained parameters and train as few additional parameters as possible. Adapter modules (Houlsby et al., 2019; Lin et al., 2020) insert additional multilayer perceptrons between layers of pretrained language models. Prefix-tuning stores much fewer parameters (e.g. 250K for table-to-text) than the adaptor module (around 10M). Additionally, prefix-tuning has a more modular architecture, leading to engineering benefits in batching across tasks (§8).

We evaluate prefix-tuning on table-to-text generation using GPT-2 and abstractive summarization using BART. On the full datasets, prefix-tuning and fine-tuning are comparable for table-to-text, while suffers small degradation for summarization. In low-data settings, prefix-tuning on average outperforms fine-tuning in both tasks (§ 6.3). Prefix-tuning also extrapolates better to table contents and summarization documents of unseen topics (§ 6.4).

2 Related Work

Fine-Tuning for Natural Language Generation.

State-of-the-art performance on many NLG benchmarks are obtained by fine-tuning. For table-to-text generation, Kale (2020) fine-tune pretrained a sequence-to-sequence model (Raffel et al., 2020) to achieve SOTA performance on multiple benchmarks. Competitive extractive and abstractive summarization systems are obtained by fine-tuning masked language models (e.g. BERT) or encode-decoder models (e.g. BART) respectively (Zhong et al., 2020; Liu and Lapata, 2019; Raffel et al., 2020; Lewis et al., 2020). For other conditional NLG tasks such as machine translation or dialogue generation, pretraining and fine-tuning are also the prevalent paradigm (Zhang et al., 2020; Stickland et al., 2020; Zhu et al., 2020; Liu et al., 2020). In this paper, we focus on summarization and table-to-text, but prefix-tuning can be easily generalizable to other generation tasks and model architectures.

Lightweight fine-tuning. Prefix-tuning is a form of lightweight fine-tuning, the core of which is to explore the right architectures. One line of research freezes the pretrained parameters and attaches small trainable networks to the large pretrained model. For example, Zhang et al. (2020) trains a “side” network that is fused with the pretrained network via summation, and Houlsby et al. (2019) inserts trainable adapters between each layer

of the pretrained transformer models. Another line of research ablates away some model weights by training a binary mask over model parameters to decide which ones to keep (Zhao et al., 2020). Prefix-tuning differs from previous methods in that it makes no modification to the model architecture. This modularity is not only conceptually favorable but also provides additional engineering benefits when batching across tasks (§8).

Prompting. Prefix-tuning is a form of continuous prompting. In addition to previously mentioned work on discrete prompting to improve classification or knowledge extraction accuracy, researchers also use manually designed keywords as prompts to control for sentiment or topic of the generated sentence (Sun and Lai, 2020). Continuous prompts are more expressive and easier to train than its discrete counterparts. Although GPT-3 can be effectively prompted by natural language task instructions and a few examples, our method could ideally be extended to GPT-3 (assume access to gradient information) and the learned continuous prompt is likely to outperform the human written discrete prompt. One earlier attempt to condition a language model by a continuous sentence vector is the work of Subramani et al. (2020), who showed empirically that the LSTM language model can recover arbitrary sentences by selecting a good conditioning vector. For each sentence input, they optimize a vector to reconstruct this particular sentence, and therefore, a different sentence corresponds to a different vector. Prefix-tuning differs in that it learns a task-specific prefix and all the instances of the task share the same prefix, enabling prefix-tuning to do NLG tasks and have fast inference.

Controllable Generation. Controllable generation aims to steer a pretrained language model to match a sentence level attribute (e.g. positive sentiment or topic on sports). Such control can happen at training time: Keskar et al. (2019) pretrains the language model (CTRL) to condition on meta data like keywords or URLs. Additionally, the control can happen at decoding time, by weighted deencoding (Krause et al., 2020) or iteratively updating the past activations (Dathathri et al., 2020). Despite the success of controlling sentence attributes, these controllable generation techniques are not sufficient for fine-grained control over contents, as demanded by some NLG tasks like table-to-text and summarization.

3 Problem Statement

Consider a conditional generation setting where the input consists of an arbitrary context x and the output y is a sequence of target tokens. The goal is to generate y conditioned on x by learning a model of $p(y | x)$. Concretely, we focus on two tasks: (1) In a table-to-text problem, x corresponds to a linearized table of data and y is a textual description; (2) In a summarization problem, x is an article and y is a short summary.

3.1 Notation

Assume we have an auto-regressive language model based on a transformer architecture (e.g. GPT-2). Since this model does not have an explicit encoder to encode x , we put x in the left context of y and obtain this concatenation $z = [x; y]$. We assign indices to different chunks in z : X_{idx} denotes the sequence of indices that corresponds to x , and Y_{idx} denotes the same for y , as shown in the upper part of Figure 2. The activation at time step t is $h_t \in \mathbf{R}^d$, where $h_t = [h_t^{(1)}; \dots; h_t^{(n)}]$ is a concatenation of all activation layers at this time step². Due to the autoregressive property of this language model, h_t is computed deterministically given the t th token and all the tokens to its left.

We can also approach this conditional generation task by a transformer-based encoder-decoder architecture (e.g. BART) where x is encoded by the encoder, and the decoder predicts y autoregressively (conditioned on the encoded x and its left context). We use the same indexing and activation notation, as shown in the bottom part of Figure 2.

3.2 Method: Fine-Tuning

The recurrence relation of an autoregressive transformer model computes h_i as a function of z_i and the past activations in its left context.

$$o_{i+1}, h_i = \text{LM}_\phi(z_i, h_{<i}) \quad (1)$$

where o_{i+1} is used to compute the distribution for the next token: $p_\phi(z_{i+1} | z_{\leq i}) = \text{softmax}(W_\phi \cdot o_{i+1})$ and W_ϕ is some pretrained parameters that maps o_i to logits over vocabularies. In the fine-tuning framework, p_ϕ is a trainable language model distribution parametrized by ϕ and the objective is

² $h_t^{(i)}$ denotes the activation of the i th layer at time step t , composed of a key-value pair. In GPT-2, the dimension of each key and value is 1024.

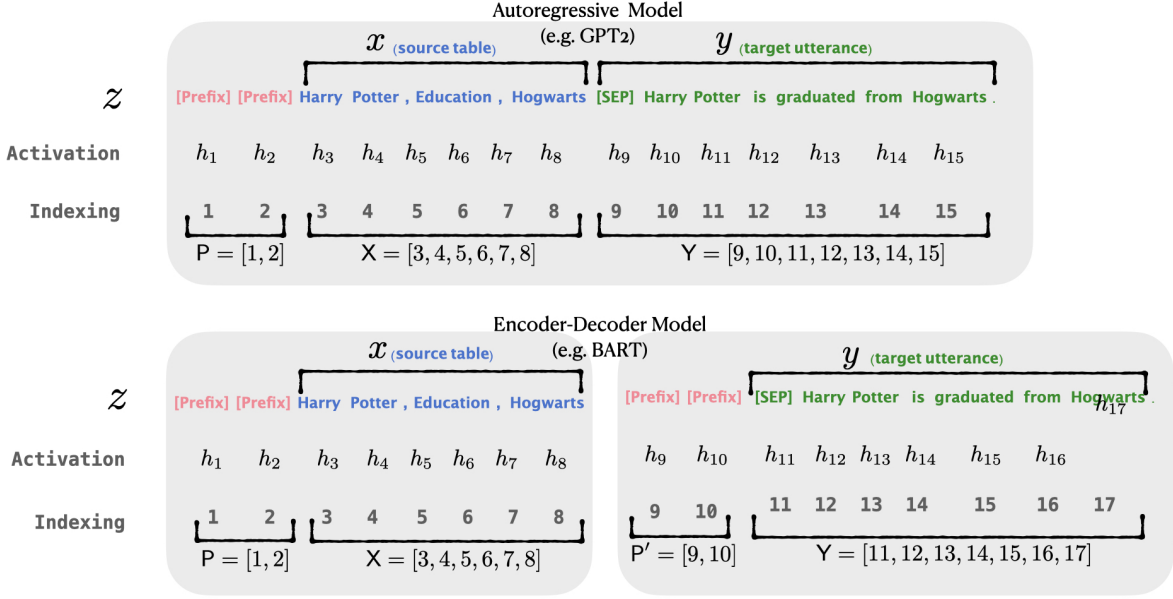


Figure 2: An annotated example of prefix-tuning using autoregressive LM (top) and encoder-decoder (bottom). The prefix activations $h_i \forall i \in P_{\text{idx}}$ are drawn from a trainable LOOKUP_θ table. The remaining activations are computed by the transformer.

to maximize the following log-likelihood.

$$\max_{\phi} \log p_{\phi}(y | x) = \sum_{i \in Y_{\text{idx}}} \log p_{\phi}(z_i | h_{<i}) \quad (2)$$

4 Prefix-Tuning

Prefix-tuning prepends a prefix of length $|P_{\text{idx}}|$ to obtain $z = [\text{PREFIX}; x; y]$, as shown in Figure 2. Here, P_{idx} denotes the sequence of prefix indices. We follow the recurrence relation in equation (1), except that prefix activations are *directly intervened*. Prefix-tuning initializes a trainable lookup table (parametrized by θ) to store the parameters for the prefix activations.

$$h_i = \begin{cases} \text{LOOKUP}_{\theta}(i), & \text{if } i \in P_{\text{idx}} \\ \text{LM}(z_i, h_{<i}), & \text{otherwise} \end{cases} \quad (3)$$

The training objective is the same as equation (2), but the set of trainable parameters changes. LM parameters are fixed and h_i (for all i) is a function of the trainable LOOKUP_{θ} . If $i \in P_{\text{idx}}$, this is obvious because h_i copies from the lookup table. For $i \notin P_{\text{idx}}$, h_i still depends on the lookup table, because the prefix activations are always in the left context and therefore will affect any activations to its right.

4.1 Parametrization Trick

Empirically, directly optimizing the lookup table parameters leads to suboptimal performances. So

we reparametrize the lookup table $\text{LOOKUP}_{\theta}(i) = \text{MLP}_{\theta}(\text{LOOKUP}'_{\theta}(i))$ by a smaller lookup table (LOOKUP'_{θ}) composed with a feedforward neural network (MLP_{θ}). The LOOKUP_{θ} and LOOKUP'_{θ} have the same number of entries (i.e. the prefix length) but LOOKUP'_{θ} has a much smaller dimension³. Reparametrizing the lookup table makes training easier by introducing more trainable parameters (e.g. 25M for table-to-text) and an MLP architecture that's easier to optimize. Once training is complete, these reparametrization parameters can be dropped, and only the prefix (stored in LOOKUP_{θ}) needs to be saved.

5 Experimental Setup

5.1 Datasets and Metrics

We consider three standard neural generation benchmarks for the table-to-text tasks: E2E (Novikova et al., 2017), WebNLG (Gardent et al., 2017), and DART (Radev et al., 2020), with examples shown in ???. The E2E dataset contains approximately 50K examples with 8 distinct fields; it contains multiple test references for one source table. We evaluate in terms of BLEU (Papineni et al., 2002), NIST (Belz and Reiter, 2006), METEOR (Lavie and Agarwal, 2007), ROUGE-L (Lin,

³ LOOKUP_{θ} has a dimension of $\dim(h_i)$ while the dimension of LOOKUP'_{θ} is a hyper-parameter. MLP_{θ} maps a vector with the same dimension as LOOKUP'_{θ} to a vector of $\dim(h_i)$

E2E						WebNLG									
	BLEU	NIST	MET	ROUGE	CIDEr	BLEU \uparrow			METEOR \uparrow			TER \downarrow			
						SEEN	UNSEEN	ALL	SEEN	UNSEEN	ALL	SEEN	UNSEEN	ALL	
GPT2-medium						GPT2-medium									
Fine-tune	68.21	8.62	46.2	71.0	2.47	Fine-tune	64.24	27.71	46.52	0.45	0.30	0.38	0.33	0.76	0.53
FT (top 1)	67.54	8.59	44.2	70.0	2.26	FT(top 2)	53.56	18.92	36.00	0.38	0.23	0.31	0.49	0.99	0.72
SOTA	68.60	8.73	45.3	70.8	2.37	Adapter	56.01	46.25	51.58	0.40	0.36	0.39	0.38	0.46	0.42
Adapter	66.56	8.51	44.2	69.3	2.34	Prefix	63.37	44.88	55.01	0.44	0.38	0.41	0.34	0.49	0.41
Prefix	69.65	8.76	45.7	71.3	2.43	SOTA	63.90	52.80	57.10	0.46	0.41	0.44	-	-	-
GPT2-large						GPT2-large									
Fine-tune	68.52	8.78	46.0	69.9	2.45	Fine-tune	65.30	43.12	55.50	0.46	0.38	0.42	0.33	0.53	0.42
Prefix	70.31	8.85	46.2	71.7	2.47	Prefix	63.36	47.67	56.25	0.45	0.39	0.42	0.34	0.48	0.40

Figure 3: Automatic metrics for text generation on E2E (left) and WebNLG (right).

2004), CIDEr (Vedantam et al., 2015), using the official evaluation scripts.

The WebNLG (Gardent et al., 2017) dataset consists of 22K examples, and the input x is a sequence of (subject, property, object) triples. In the training and validation splits, the input describes entities from 9 distinct DBpedia categories. The test split consists of two parts: the first half contains DB categories seen in training data, and the second half contains 5 unseen categories. These unseen categories are used to evaluate extrapolation performance. We use the official evaluation scripts, which reports BLEU, METEOR and TER (Snover et al., 2006).

DART (Radev et al., 2020) is an open domain table-to-text dataset, with similar input format (entity-relation triples) as WebNLG. It consists of 82K examples from WikiSQL, WikiTableQuestions, E2E, and WebNLG with some manual or automated conversion. We evaluate in terms of BLEU, METEOR, TER, MoverScore (Zhao et al., 2019), BERTScore (Zhang* et al., 2020) and BELURT (Sellam et al., 2020).

For the summarization task, we focus on the XSUM (Narayan et al., 2018) dataset, which is an abstractive summarization dataset on news articles. There are 225K examples. The average length of the articles is 431 words and the average length of the summaries is 23.3. We evaluate based on ROUGE-1, ROUGE-2 and ROUGE-L.

5.2 Baselines

For table-to-text generation, we compare externally against three other methods: fine-tuning, fine-tuning only the top 2 layers, and adapter modules. We also report state-of-the-art results on these datasets respectively: on E2E, Shen et al. (2019) uses a pragmatically informed model. On WebNLG, Kale (2020) fine-tunes T5-large. On DART, a comparable benchmark model is not avail-

able⁴. For summarization, we compare against fine-tuning BART (Lewis et al., 2020).

For intrinsic evaluation, we compare different variants of prefix-tuning. § 7.1 studies prefix-tuning with different prefix length. § 7.2 ablates the prefix to optimize only in the embedding level. § 7.3 compares prefixing and infixing, which inserts trainable activations between x and y . § 7.4 studies initialization techniques of the prefix.

5.3 Architecture and Hyperparameters

For table-to-text generation, we use (medium and large) GPT-2 as our base model and the source tables are linearized (details in Appendix A). For summarization, we use BART-large as our base model⁵, and the source documents are truncated to 512 BPE tokens.

Our implementation is based on the Huggingface transformer models (Wolf et al., 2020). At training time, we use the AdamW optimizer (Loshchilov and Hutter, 2019) and a linear learning rate scheduler, as suggested by the HuggingFace default setup. The hyperparameters we tune include the learning rate, prefix length, number of epochs, batch size, and dimension of the small look up table. Hyperparameter details are in the appendix.

At decoding time for the three table-to-text datasets, we use beam search with a beam size of 5. For summarization, we use a beam size of 6 with length normalization as a hyperparameter.

6 Main Results

6.1 Table-to-text Generation

On E2E (Figure 3, left), prefix-tuning achieves results comparable or better than fine-tuning for both

⁴The official benchmark model is trained on v.1.0.0 while the release dataset is v1.1.1.

⁵We didn’t include GPT-2 results for summarization because in our preliminary experiment, fine-tuning GPT-2 significantly underperforms fine-tuning BART.

	BLEU	MET	TER↓	Mover	BERT	BLEURT
GPT2-medium						
Fine-tune	46.22	0.39	0.46	0.50	0.94	0.39
FT (top 2)	40.98	0.34	0.56	0.43	0.93	0.21
Adapter	43.89	0.37	0.47	0.49	0.94	0.37
Prefix	46.34	0.38	0.46	0.51	0.94	0.39
GPT2-large						
Fine-tune	46.96	0.39	0.46	0.51	0.94	0.40
Prefix	46.65	0.39	0.45	0.51	0.94	0.40

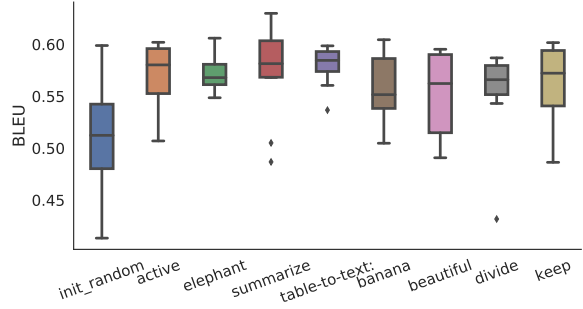


Figure 4: Automatic metrics for text generation on DART (left). Initialization trick in low-data settings (right).

GPT-2 medium and large. It also has significantly better performance than the adapter modules and fine-tuning only the top two layers. This demonstrates that prefix-tuning is sufficiently expressive to steer a pretrained LM to specialize in a simple table-to-text setting.

On WebNLG (Figure 3, right), prefix-tuning outperforms fine-tuning and all the other baselines on the “ALL” column, which is a combination of “SEEN” and “UNSEEN” categories. For the “SEEN” column, prefix-tuning is slightly worse than fine-tuning across all three metrics, but outperforms other lightweight baselines including fine-tuning the top 2 layers and the adapter module. For the “UNSEEN” column, prefix-tuning achieves significantly better results than fine-tuning across all metrics. Having a slightly better performance than adapter on one metric and slightly worse on the other two. We will discuss the “UNSEEN” performance in details in § 6.4. The performance in WebNLG dataset indicates that prefix-tuning achieves a good tradeoff between the “SEEN” and “UNSEEN” categories.

On DART (Figure 4 left), prefix-tuning outperforms fine-tuning on two metrics and remain comparable on the rest. It also achieves better performance than the adapter module and finetuning (top2) across all metrics. Good performance here suggests that prefix-tuning works well with open domain tables and a large pool of diverse relations.

Overall, the results show that prefix-tuning can effectively steer a pretrained GPT-2 model for table-to-text generation, with a comparable performance to fine-tuning and better performance than other lightweight baselines.

6.2 Summarization

	R-1 ↑	R-2 ↑	R-L ↑
Fine-tune (Lewis et al., 2020)	45.14	22.27	37.25
Prefix	43.57	20.60	35.70

The table above shows the main result for summarization on the XSUM dataset, where prefix-tuning underperforms the fine-tuning baseline. This summarization dataset has more than twice the number of training examples than previously mentioned table-to-text datasets. Therefore, under such a high resource setting, prefix-tuning is not favorable if the ultimate goal is to saturate the model to fit the data, however, it could still be favorable from the perspective of a light storage.

6.3 Low-data Setting

To construct low-data settings, we randomly sample data from the full dataset (E2E for table-to-text and XSUM for summarization) to obtain small datasets of size $\{50, 100, 200, 500\}$. For each size, we sample 5 different datasets and average over 2 training random seeds. Thus, we average over 10 models to get a reliable estimate for each low-data setting⁶. In Figure 5 (right), we compare prefix-tuning and fine-tuning for different data sizes. As the number of training examples increases, both prefix-tuning and fine-tuning show a trend of increasing performance, as expected. Notably, prefix-tuning consistently outperforms fine-tuning for both summarization and table-to-text.

For qualitative comparison, Figure 5 (left) shows 8 examples generated by table-to-text models trained on different data levels. When training on a small amount of data, the generated sentences tend to have a low coverage and ignore some table

⁶We also sample a dev split for each training set, containing 30% of examples as the training size. We use the dev split to choose hyper-parameters and do early stopping.

Source	name : The Eagle type : coffee shop food : Chinese price : cheap customer rating : average area : riverside family friendly : no near : Burger King
Prefix (50)	The Eagle is a cheap Chinese coffee shop located near Burger King.
Prefix (100)	The Eagle is a cheap coffee shop located in the riverside near Burger King. It has average customer ratings.
Prefix (200)	The Eagle is a cheap Chinese coffee shop located in the riverside area near Burger King. It has average customer ratings.
Prefix (500)	The Eagle is a coffee shop that serves Chinese food. It is located in the riverside area near Burger King. It has an average customer rating and is not family friendly.
FT (50)	The Eagle coffee shop is located in the riverside area near Burger King.
FT (100)	The Eagle is a cheap coffee shop near Burger King in the riverside area. It has a low customer rating and is not family friendly.
FT (200)	The Eagle is a cheap Chinese coffee shop with a low customer rating. It is located near Burger King in the riverside area.
FT (500)	The Eagle is a cheap Chinese coffee shop with average customer ratings. It is located in the riverside area near Burger King.

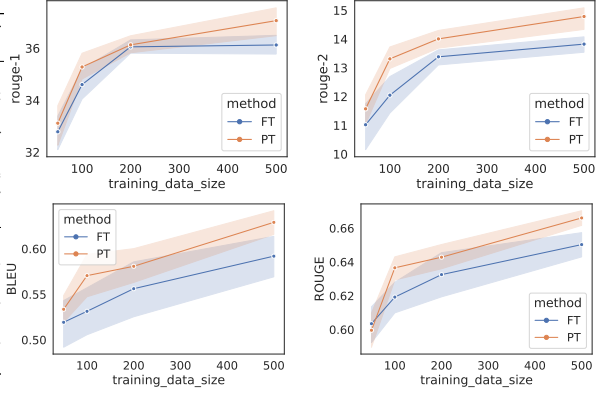


Figure 5: Qualitative examples in lowdata settings (left). Low-data performance (right) of prefix-tuning (PT in orange) and fine-tuning (FT in blue). The top two plots correspond to summarization, measured by ROUGE-1 and ROUGE-2, respectively. The bottom two plots correspond to table-to-text generation, measured by BLEU and ROUGE-L. The x-axis is the training size and the y-axis is the evaluation metric (higher is better).

contents. This happens for both prefix-tuning and fine-tuning. Models also hallucinate and generate contents that are not faithful to the table, especially in the case of fine-tuning (e.g. Fine-tuning (100, 200) in Figure 5 falsely claim a low customer rating while the true rating is average). As the number of training examples increases, the generated sentences tend to have a higher coverage and be more truthful.

6.4 Extrapolation

We study the extrapolation performance to unseen topics for both table-to-text and summarization. In the table-to-text setting, we use the “SEEN” and “UNSEEN” categories in WebNLG as the standard dataset split. The training and validation split only contains the “SEEN” categories (e.g. Comics), and those unseen categories (e.g. Athlete) appear only at test time. As shown in the “UNSEEN” column of Figure 3, prefix-tuning has significantly better extrapolation than fine-tuning under all metrics. Additionally, scaling up from GPT-2 medium to large boosts the extrapolation performance for both fine-tuning and prefix-tuning. Although prefix-tuning still performs much better than fine-tuning using GPT-2 large, the performance gap shrinks compared with GPT-2 medium.

	News-to-Sport			Within News		
	R-1 ↑	R-2 ↑	R-L ↑	R-1 ↑	R-2 ↑	R-L ↑
Fine-tune	38.15	15.51	30.26	39.20	16.35	31.15
Prefix	39.23	16.74	31.51	39.41	16.87	31.47

For summarization, we construct two data splits to evaluate extrapolation. In the first split (news-to-sports), the training and validation sets only include news articles, but we use sports articles to test. In the second split (within-news), the training and

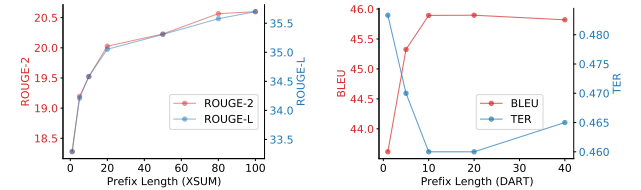


Figure 6: Generation performance vs. prefix length on summarization (left) and table-to-text (right). Each plot has two vertical axis, representing different metrics. We report ROUGE-2 (red) and ROUGE-L (blue) for summarization; BLEU (red) and TER (blue) for table-to-text. Additional plots for all datasets and metrics are in Appendix.

validation sets only include {world, uk, business} news, and we use the remaining news categories to test (e.g. health and technology). We believe the news-to-sport extrapolation to be harder since news and sports have more differences in terms of keywords and writing styles. As shown in the table above, prefix-tuning outperforms fine-tuning in both data splits. Interestingly, the performance gap is larger for the news-to-sports split, suggesting that prefix-tuning has more advantages when the extrapolation becomes harder.

7 Intrinsic Evaluation

7.1 Prefix Length

Figure 6 plots the relations between prefix length and generation performance. For summarization task, as prefix length increases, we can see a performance gain. For table-to-text (DART) generation, when prefix length < 10, increasing it helps the performance (note that TER is the lower the better). There are diminishing returns for prefix length

Source	(Genoa, location, Costa Crociere), (AIDA Cruises, operator, AIDAstella), (Costa Crociere, owner, AIDAstella)						
Prefix	AID Astella is operated by Aida Cruises and is owned by the Costa Rican tourist resort of Genoa.	E2E					
		BLEU	NIST	MET	ROUGE	CIDEr	
Fine-tuning	AID Astella, operated by AIDA-Cruises, is located in Genoa and is owned by the Costa Rican government.	PT (Full)	69.65	8.76	45.7	71.3	2.43
Reference	Costa Crociere is the owner of the AIDAstella and are based in Genoa. The operator of AIDAstella is AIDA Cruises.	Embedding Ablation Studies					
		Emb-1	48.05	3.33	32.1	60.2	1.10
		Emb-5	57.08	5.22	36.3	64.3	1.49
		Emb-10	62.16	6.70	38.6	66.4	1.75
		Emb-20	61.90	7.11	39.3	65.6	1.85
Source	(Americans, nationality, Ducan Rouleau), (Ducan Rouleau, creator, Baymax),(Alan Tudyk, starring, Big Hero 6 (film)), (Steven T Segle, creator, Baymax), (Big Hero 6 (film), series, Baymax)						
Prefix	Baymax is a character in Big Hero 6 which stars Alan Tudyk. He was created by Steven T. Seagle and the American, Duncan Rouleau.	Infix Ablation Studies					
Fine-tuning	Alan Tudyk stars in the film Big Hero 6 in which Baymax is a character created by Steven T. Seagle and the American, Duncan Rouleau.	Infix-1	67.89	8.63	45.8	69.4	2.42
		Infix-10	67.21	8.48	45.8	69.9	2.40
Reference	Baymax is a character who appeared in Big Hero 6 starring Alan Tudyk. It was created by Steven T Seagle and the American, Duncan Rouleau.	Infix-20	66.70	8.47	45.8	70.0	2.42

Table 1: WebNLG qualitative examples. The source table is a sequence of (object, property, subject) triples. Ablation studies results(right)

> 10, suggesting that 10 is expressive enough for this table-to-text task, and having longer prefixes is prone to overfitting the training data ⁷.

7.2 Full vs Embedding-only

In this ablation study, we only optimize the word embedding space of the prefix, (i.e. the 0th layer of the full activation space) and use the pretrained LM parameters to deterministically compute the remaining activation layers bottom up. Because this optimization is restricted only to the 0th layer of activation, the performance dropped significantly, as shown in the top part of Table 1, suggesting that embedding-only prefix is not sufficiently expressive. Also, we discover that a longer prefix here leads to performance gain, (contrast with the diminishing return in full prefix-tuning) because the embedding-only version lacks expressiveness and longer prefix alleviates this problem.

7.3 Prefixing vs Infixing

Due to the autoregressive nature of language models, placing the trainable activations at the very beginning [PREFIX; x ; y] means that the activations of both input x and output y will be affected. Infixing inserts the trainable activation layer between x and y , [x ; INFIX; y]. As a result, only y 's activation will be affected, leading to slightly weaker performance than prefix-tuning, as shown in the bottom part of Table 1.

⁷The average negative log likelihood (training loss) is 21.78 for prefix length of 20, and 22.78 for prefix length of 10, suggesting the longer prefix overfits the training split.

7.4 Initialization

We discover that initializing the prefix is instrumental to prefix-tuning in low-data settings. Random initialization⁸ leads to low performances with a high variance. Initializing the prefix with activations of real words significantly improves generation, as shown in Figure 4 (right). What word then should we use to initialize the prefix? We experiment with task related words, such as "summarization" and "table-to-text" as well as irrelevant words, such as "elephant", "divide" and "active". We find that task relevant words obtain a slightly better performance than task-irrelevant words, but both are significantly better than random initialization.

8 Discussion

Prefix-tuning is helpful for research developing, when experimenting and checkpointing different model variants and hyperparameters. In this section, we will focus on the modularity of prefix-tuning and how it can benefit deployment.

8.1 Modularity of Prefix-Tuning

Adding or Deleting Tasks As we note in §1, prefix-tuning only stores a small prefix for each additional task, obtaining not only storage benefits, but also favorable *modularity across tasks*. To compare with fine-tuning LM with a multi-task objective, which has the same storage benefits by storing a single LM for k tasks, we consider two cases: adding new tasks and deleting old tasks.

⁸from a uniform(0,1) distribution

Both are challenging for multi-task LM but trivial for prefix-tuning, thanks to its modularity.

Imagine adding a new task to a trained multi-task LM, fine-tuning only on the new task might weaken the performance for the old tasks whereas fine-tuning on $k + 1$ tasks jointly involves retrieving data for the old tasks. For prefix-tuning, adding a new task only involves learning a new prefix. Additionally, if we need to delete a task (e.g. for ethical concerns), the multi-task LM needs to be retrained on the $k - 1$ tasks in order to “forget” about this one task. For prefix-tuning, deleting the task simply corresponds to deleting that prefix.

Note that this modular property across task is not unique to prefix-tuning; it also applies to other lightweight fine-tuning approaches that freeze the pretrained model and attaches additional trainable modules.

User Privacy. Prefix-tuning is a efficient way to preserve user privacy, thanks to its *modularity across users*⁹. Suppose we want to train a personalized text completion system for each user using their private data. A multi-task approach would mix data for different users to jointly train a model; however, this might disclose private user information. Fine-tuning a personalized model for each user preserves user privacy, but is prohibitively expensive in storage (either the edge device has to store a large model, or the cloud device has to store millions of big models, depending on the number of users). Prefix-tuning (and other lightweight fine-tuning approaches) preserves privacy in a storage efficient way, and thus might be deployable in settings where personalization and privacy matter.

Batching Benefits. Prefix-tuning has a very modular architecture by not touching the LM architecture or its parameters, unlike other lightweight fine-tuning approaches which mask out some LM weights (Zhao et al., 2020) or inserts modules between transformer layers (Houlsby et al., 2019). Prefix-tuning obtains some engineering benefits in batching across tasks. Imagine translating a English sentence to 10 other languages, instead of querying 10 times, batching these 10 queries could speed up translation. Translation models with adapters have shared transformer layers but different adapter modules, therefore, batching requires engineering efforts to assign examples to

⁹This is a subcase of modularity across tasks, where we treat each user as a separate task whose personal data and model are not mixed together.

different adapters. Prefix-tuning can batch across tasks with little additional effort: we just need to prepend different prefixes to each example in the data processing stage.

8.2 Prefix-Tuning as Probing

1. Does prefix-tuning (or lightweight fine-tuning) count as probing? based on the current probing definition.

2. The shallow v.s. deep contrast with previous probing method...

9 Conclusion and Future works

In this paper, we have proposed prefix-tuning, a lightweight alternative to fine-tuning that prepends a trainable continuous prompt for NLG tasks. We discover that despite only storing 1000x fewer parameters than fine-tuning, prefix-tuning can maintain a comparable performance in a full data setting and outperforms fine-tuning in both low-data and extrapolation settings.

One could apply the same prefix-tuning method for other classification or generation tasks. All that is required is the main model has a transformer backbone, which applies to all large pretrained models in NLP. For example, in this case of sentiment analysis using BERT, the prefix can be concatenated with the input sentence and optimized for classification accuracy. Additionally, performance gains in extrapolation setting is interesting but underexplored. We hypothesize that freezing the LM parameters leads to a favorable implicit regularization and leave more exploration for future work.

References

- Anja Belz and Ehud Reiter. 2006. [Comparing automatic and human evaluation of NLG systems](#). In *11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy. Association for Computational Linguistics.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. [Language models are few-shot learners](#).

- Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. [Plug and play language models: A simple approach to controlled text generation](#). In *International Conference on Learning Representations*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). 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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. [Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping](#).
- Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. [The WebNLG challenge: Generating text from RDF data](#). In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133, Santiago de Compostela, Spain. Association for Computational Linguistics.
- Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. [Parameter-efficient transfer learning for NLP](#). In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799, Long Beach, California, USA. PMLR.
- Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. [How can we know what language models know?](#) *Transactions of the Association for Computational Linguistics*, 8:423–438.
- Mihir Kale. 2020. [Text-to-text pre-training for data-to-text tasks](#).
- N. Keskar, B. McCann, L. R. Varshney, Caiming Xiong, and R. Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *ArXiv*, abs/1909.05858.
- Simon Kornblith, Jonathon Shlens, and Quoc V. Le. 2019. Do better imagenet models transfer better? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. 2020. GeDi: Generative Discriminator Guided Sequence Generation. *arXiv preprint arXiv:2009.06367*.
- 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, StatMT '07*, pages 228–231, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. [BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.
- Chin-Yew Lin. 2004. [ROUGE: A package for automatic evaluation of summaries](#). In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.
- Zhaojiang Lin, Andrea Madotto, and Pascale Fung. 2020. [Exploring versatile generative language model via parameter-efficient transfer learning](#). In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 441–459, Online. Association for Computational Linguistics.
- Yang Liu and Mirella Lapata. 2019. [Text summarization with pretrained encoders](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3730–3740, Hong Kong, China. Association for Computational Linguistics.
- Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. [Multilingual denoising pre-training for neural machine translation](#).
- Ilya Loshchilov and Frank Hutter. 2019. [Decoupled weight decay regularization](#). In *International Conference on Learning Representations*.
- Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don’t give me the details, just the summary! Topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium.
- Jekaterina Novikova, Ondrej Dusek, and Verena Rieser. 2017. [The E2E dataset: New challenges for end-to-end generation](#). *CoRR*, abs/1706.09254.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. [Bleu: A method for automatic evaluation of machine translation](#). In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, ACL '02*, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

- Dragomir Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Nazneen Fatema Rajani, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, Yangxiaokang Liu, Nadia Irwanto, Jessica Pan, Faiaz Rahman, Ahmad Zaidi, Murori Mutuma, Yasin Tarabar, Ankit Gupta, Tao Yu, Yi Chern Tan, Xi Victoria Lin, Caiming Xiong, and Richard Socher. 2020. [Dart: Open-domain structured data record to text generation](#).
- A. Radford, Jeffrey Wu, R. Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.
- 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](#). *Journal of Machine Learning Research*, 21(140):1–67.
- Timo Schick and Hinrich Schütze. 2020. [Exploiting cloze questions for few shot text classification and natural language inference](#).
- Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. [BLEURT: Learning robust metrics for text generation](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.
- Sheng Shen, Daniel Fried, Jacob Andreas, and Dan Klein. 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)*, pages 4060–4067, Minneapolis, Minnesota. Association for Computational Linguistics.
- Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. [Auto-prompt: Eliciting knowledge from language models with automatically generated prompts](#).
- Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and Ralph Weischedel. 2006. A study of translation error rate with targeted human annotation. In *Proceedings of the Association for Machine Translation in the Americas (AMTA 2006)*.
- Asa Cooper Stickland, Xian Li, and Marjan Ghazvininejad. 2020. [Recipes for adapting pre-trained monolingual and multilingual models to machine translation](#).
- Nishant Subramani, Samuel R. Bowman, and Kyunghyun Cho. 2020. [Can unconditional language models recover arbitrary sentences?](#)
- Fan-Keng Sun and Cheng-I Lai. 2020. [Conditioned natural language generation using only unconditioned language model: An exploration](#).
- Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. 2015. [Cider: Consensus-based image description evaluation](#). In *CVPR*, pages 4566–4575. IEEE Computer Society.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. [Transformers: State-of-the-art natural language processing](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.
- Jeffrey O Zhang, Alexander Sax, Amir Zamir, Leonidas Guibas, and Jitendra Malik. 2020. [Side-tuning: A baseline for network adaptation via additive side networks](#).
- Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. [Bertscore: Evaluating text generation with bert](#). In *International Conference on Learning Representations*.
- Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2020. [DIALOGPT : Large-scale generative pre-training for conversational response generation](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 270–278, Online. Association for Computational Linguistics.
- Mengjie Zhao, Tao Lin, Fei Mi, Martin Jaggi, and Hinrich Schütze. 2020. [Masking as an efficient alternative to finetuning for pretrained language models](#).
- Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M. Meyer, and Steffen Eger. 2019. [MoverScore: Text generation evaluating with contextualized embeddings and earth mover distance](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 563–578, Hong Kong, China. Association for Computational Linguistics.
- Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2020. [Extractive summarization as text matching](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6197–6208, Online. Association for Computational Linguistics.
- Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tieyan Liu. 2020. [Incorporating bert into neural machine translation](#). In *International Conference on Learning Representations*.

A Appendices

A.1 Additional Results for the Low-data Setting

A.2 Additional Results for the Initialization Experiment

	BLEU	NIST	E2E MET	ROUGE	CIDEr
50 (Finetune)	0.5192 ± 0.0576	4.2259 ± 1.4664	0.3308 ± 0.0214	0.6034 ± 0.0242	1.3197 ± 0.2136
50 (Prefix tune)	0.5334 ± 0.0349	5.3897 ± 1.1057	0.3421 ± 0.0171	0.5995 ± 0.0235	1.3989 ± 0.2294
100 (Finetune)	0.5312 ± 0.0558	3.8025 ± 1.6402	0.337 ± 0.0213	0.6191 ± 0.02	1.3252 ± 0.2399
100 (Prefix tune)	0.5794 ± 0.0182	5.6218 ± 1.0506	0.3588 ± 0.0129	0.6341 ± 0.0133	1.5737 ± 0.1558
200 (Finetune)	0.556 ± 0.0691	4.8462 ± 1.8381	0.3604 ± 0.0302	0.6324 ± 0.0314	1.5226 ± 0.3297
200 (Prefix tune)	0.5807 ± 0.0433	5.6735 ± 1.3577	0.365 ± 0.0156	0.6428 ± 0.0168	1.622 ± 0.2238
500 (Finetune)	0.5918 ± 0.0488	5.4211 ± 1.514	0.369 ± 0.0201	0.6502 ± 0.0162	1.6132 ± 0.2646
500 (Prefix tune)	0.6289 ± 0.0296	6.3938 ± 1.0749	0.3815 ± 0.0161	0.666 ± 0.0098	1.8002 ± 0.1988

Table 2: low data setting.

	R-1 \uparrow	R-2 \uparrow	R-L \uparrow
50 (Finetune)	32.78 ± 1.01	11.02 ± 1.2	24.78 ± 1.13
50 (Prefix tune)	33.11 ± 1.23	11.57 ± 0.79	25.32 ± 1.07
100 (Finetune)	34.6 ± 0.89	12.04 ± 0.9	26.05 ± 0.94
100 (Prefix tune)	35.27 ± 0.77	13.31 ± 0.55	27.33 ± 0.69
200 (Finetune)	36.05 ± 0.41	13.38 ± 0.39	27.68 ± 0.3
200 (Prefix tune)	36.12 ± 0.5	14.0 ± 0.47	28.2 ± 0.57
500 (Finetune)	36.12 ± 0.54	13.82 ± 0.41	28.29 ± 0.38
500 (Prefix tune)	37.05 ± 0.8	14.78 ± 0.6	29.15 ± 0.63

Table 3: XSUM results in a low data setting.

	BLEU	NIST	E2E MET	ROUGE	CIDEr
random	0.5073 ± 0.0534	3.8918 ± 1.684	0.3319 ± 0.0252	0.6118 ± 0.0193	1.2603 ± 0.2613
keep	0.563 ± 0.0358	5.2968 ± 1.1606	0.352 ± 0.0141	0.627 ± 0.0176	1.4938 ± 0.1494
divide	0.554 ± 0.0431	5.4018 ± 1.4366	0.3491 ± 0.0152	0.6216 ± 0.0078	1.4709 ± 0.1924
beautiful	0.551 ± 0.0407	5.2774 ± 1.5582	0.3539 ± 0.0198	0.6254 ± 0.0153	1.5055 ± 0.2371
banana	0.5568 ± 0.0346	5.4102 ± 1.212	0.3538 ± 0.0162	0.6283 ± 0.0148	1.5118 ± 0.1959
active	0.5687 ± 0.0323	5.8009 ± 1.2072	0.3582 ± 0.0171	0.6305 ± 0.0123	1.565 ± 0.2078
elephant	0.5716 ± 0.0169	5.8913 ± 0.8037	0.3582 ± 0.0102	0.631 ± 0.0105	1.5715 ± 0.1168
summarize	0.5727 ± 0.0432	5.933 ± 1.4168	0.3622 ± 0.0192	0.6317 ± 0.0195	1.6089 ± 0.2269
table-to-text:	0.5794 ± 0.0182	5.6218 ± 1.0506	0.3588 ± 0.0129	0.6341 ± 0.0133	1.5737 ± 0.1558

Table 4: E2E initialization in a low data setting.

Source [Unseen, Athlete]	(Al Kharaitiyat SC, club, Alaa Abdul-Zahra), (Al Khor, ground, Al Kharaitiyat SC), (Shabab Al-Ordon Club, club, Alaa Abdul-Zahra) (Amar Osim, manager, Al Kharaitiyat SC)
Prefix-tuning	Al Kharaitiyat SC are managed by Amar Osim and play at their ground at Al Khor. Al Kharaitiyat SC are also the club for which Alaa Abdul-Zahra is a player.
Fine-tuning	Alaa Abdul-Zahra plays for Al-Kharaitiyat SC and Shabab Al-Ordon Club. He also plays for Al-Khor and manages Al-Kharaitiyat SC.
Reference	Alaa Abdul Zahra plays for Al Kharaitiyat SC which is located at Al Khor and managed by Amar Osim. The Shabab Al-Ordon club is associated with Alaa Abdul-Zahra.
Source [Unseen, Transportation]	(Genoa, location, Costa Crociere), (AIDA Cruises, operator, AIDAstella), (Costa Crociere, owner, AIDAstella)
Prefix-tuning	AID Astella is operated by Aida Cruises and is owned by the Costa Rican tourist resort of Genoa.
Fine-tuning	AID Astella, operated by AIDA-Cruises, is located in Genoa and is owned by the Costa Rican government.
Reference	Costa Crociere is the owner of the AIDAstella and are based in Genoa. The operator of AIDAstella is AIDA Cruises.
Source [Unseen, Politician]	(Euro, currency, Netherlands), (Stellendam, birthPlace, Ab Klink), (Netherlands, nationality, Ab Klink)
Prefix-tuning	Ab Klink was born in Stellendam and is a national of the Netherlands where the currency is the Euro.
Fine-tuning	Ab Klink is a national of the Netherlands where the currency is the Euro. He was born in Stellendam.
Reference	Ab Klink was born in Stellendam in the Netherlands, where the national currency is the euro.
Source [Unseen, Politician]	(Robert E. Lee, commander, Battle of Salem Church), (American Civil War, isPartOfMilitaryConflict, Battle of Salem Church), (Battle of Salem Church, battles, Aaron S. Daggett)
Prefix-tuning	Robert E. Lee was the commander of the Battle of Salem Church which was part of the military conflict in the American Civil war.
Fine-tuning	The Battle of Salem Church is part of the American Civil War and was commanded by Robert E. Lee.
Reference	Robert E Lee was a commander in the Battle of Salem Church, which was one of the military conflicts in the American Civil War. Aaron S Daggett fought in the same battle.
Source [Unseen, Artist]	(Christian alternative rock, musicSubgenre, Alternative rock), (Alternative rock, genre, Andrew White (musician))
Prefix-tuning	Andrew White is a Christian alternative rock musician.
Fine-tuning	Andrew White, a Christian alternative rocker, performs.
Reference	The musician Andrew White's genre is alternative rock, the genre which has the sub genre Christian alternative rock.
Source [Unseen, Artist]	(Hip hop music, genre, Allen Forrest), (solo singer, background, Allen Forrest)
Prefix-tuning	Allen Forrest is a solo singer.
Fine-tuning	Born in
Reference	Allen Forrest is a solo singer whose genre is Hip Hop music.
Source [Seen, ComicsCharacter]	(Americans, nationality, Duncan Rouleau), (Duncan Rouleau, creator, Baymax), (Alan Tudyk, starring, Big Hero 6 (film)), (Steven T Seagle, creator, Baymax), (Big Hero 6 (film), series, Baymax)
Prefix-tuning	Baymax is a character in Big Hero 6 which stars Alan Tudyk. He was created by Steven T. Seagle and the American, Duncan Rouleau.
Fine-tuning	Alan Tudyk stars in the film Big Hero 6 in which Baymax is a character created by Steven T. Seagle and the American, Duncan Rouleau.
Reference	Baymax is a character who appeared in Big Hero 6 starring Alan Tudyk. It was created by Steven T Seagle and the American, Duncan Rouleau.
Source [Seen, City]	(Washington, D.C., capital, United States), (White Americans, ethnicGroup, United States), (United States, country, New Jersey), (New York City, largest City, United States), (New Jersey, isPartOf, Atlantic City)
Prefix-tuning	Washington D.C. is the capital of the United States where the largest city is New York City and the White Americans are an ethnic group. Atlantic City, New Jersey is also part of the United States.
Fine-tuning	Atlantic City, New Jersey is part of New Jersey in the United States. The capital city is Washington D.C. and one of the ethnic groups is White Americans.
Reference	New York City (NYC) is the largest U.S. city. Atlantic City, New Jersey are also part of the United States with its capital as Washington, DC and home to White Americans.

Table 5: WebNLG qualitative examples. The source table is a sequence of (object, property, subject) triples.