Reimplementing Prefix-Tuning: Optimizing Continuous Prompts for Generation

Mac Turner, Michael Ngo, Eric Hu, Neeraj Parihar

May 11, 2025

github.com/prefix-tuning

1 Introduction

Finetuning LLMs for specific tasks requires training a new set of weights for every task, which is expressive but expensive. On the other hand, prompt engineering does not require any training, which is cheap but limiting. *Prefix-Tuning: Optimizing Continuous Prompts for Generation* by Xiang Lisa Li and Percy Liang [1] introduces a method of fine-tuning that combines the efficiency of prompt engineering and the effectiveness of full fine tuning.

The paper shows that prefix tuning GPT-2, BART and T5 on table-to-text generation and text summarization performs better than fine tuning and other parameter efficient fine tuning methods. Additionally, prefix tuning is more parameter efficient and thus faster to train than other fine tuning methods.

2 Chosen Result

We chose to reproduce prefix-tuning for GPT-2 Medium on table-to-text generation and compare it against full fine tuning. This is the first two rows and the first column of Table 1 in the original paper. We chose this result because it shows the major result of the paper: that prefix-tuning is comparable, if not better than fine tuning. We chose to implement prefix-tuning only with GPT-2 and the E2E table-to-text dataset, since using other models and datasets would just be reimplementing the same method using different code infrastructures.

3 Methodology

3.1 Prefix-Tuning Architecture

See Figure 1. The base model we freeze is GPT-2 Medium. The model is a stack of attention blocks with hidden size 768. For each attention block i, Prefix-Tuning learns L keys and values to attended over during generation. These L keys and values are learned via a network, $\mathsf{MLP}_{\theta,i}$ and input matrix $P_{\theta,i}$. This is done for training stability. $P_{\theta,i}$ is of dimension $L \times 768$. The MLP is 2 layers, specifically a linear layer from 768 to 800, followed by a tanh activation, and a linear layer from 800 to the embedding dimension of $2 \cdot 784$. Note there is multiplicationby 2 because for each activation, we need a key and value.

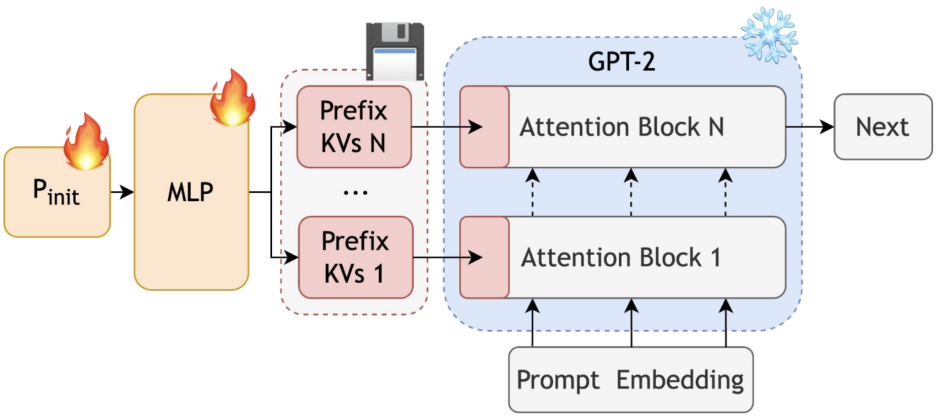

Figure 1: Prefix-Tuning Architecture

To do prefix-tuning, we load a pretrained GPT-2 Medium model from HuggingFace and pass in the learned keys and values through the `past_key_values` keyword. Additionally, generation was done via beam search with beam length of 5.

3.2 Dataset

The dataset is the E2E Table-to-text generation dataset [2]. It is a dataset of tables conaining information about restaurants and the goal is to write a sentence that summaries the tabular information. Generated or proposed sentences are compared to a list acceptable reference sentences and evaluated with a standard suite of metrics: BLEU, NIST, METEOR, ROUGE-L, and CIDEr.

3.3 Training

Following the paper, prefix-tuning is trained over the E2E dataset, with `epochs=5`, `lr=8e-5`, `batch_size=10`, `prefix_length=5`. Full GPT fine-tuning uses `epochs=5`, `lr=5e-5`, `batch_size=10`.

4 Results & Analysis

Model	BLEU	NIST	METEOR	ROUGE-L	CIDEr
Prefix-Tuning (0.1%) (our impl.)	**68.8**	**8.80**	45.7	**71.3**	2.41
Fine-Tuning (our impl.)	**68.8**	8.68	**45.8**	71.2	**2.45**
Prefix-Tuning (0.1%) [1]	**69.7**	**8.81**	46.1	**71.4**	**2.49**
Fine-Tuning [1]	68.2	8.62	**46.2**	71.0	2.47

Table 1: Results of prefix-tuning and full fine-tuning of GPT-2 Medium on the E2E table-to-text generation dataset.

See results in Table 1. We observe that our results are slightly worse than the paper's reported scores. However, the general trends we observed are the same: prefix-tuning has comparable performance to fine-tuning GPT-2, while optimizing much less (0.1%) parameters. We also performed

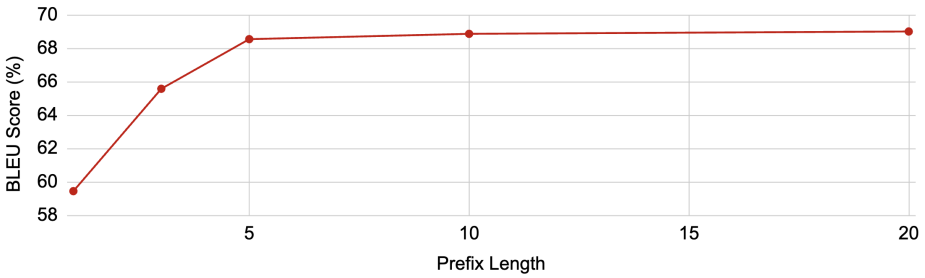

Figure 2: BLEU Score vs Prefix Length For E2E Table-to-Text Generation Task

an ablation on the prefix length, shown in Figure 2, where we found that performance stagnates with prefixes with length > 5. This is consistent with the authors' choice to use prefixes of length 5 for this task.

We faced a lot of challenges during implementation. Using the HuggingFace transformer's `generate` function produced results that omitted table information. After a lot of debugging, we decided to implement it ourselves, which includes implementing beam search from scratch. We also spent a good chunk of time figuring out how to properly pass in prefix information to the HuggingFace model.

Despite these challenges, our results align with those found the original authors: prefix tuning is about as effective as finetuning while being much more time and space efficient.

5 Reflections

We learned to iterate early and often. While understanding concepts is useful to map out potential roadblocks, it arguably more beneficial to write up code and see what actually becomes a roadblock. Prefix tuning still requires new prefixes to be trained for each task. We thought it would be interesting to have an LLM output specific tokens to utilize specific prefix weights when it needs to perform a certain task.

References

[1] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.353. URL https://aclanthology.org/2021.acl-long.353/. 1, 2

[2] Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. The E2E dataset: New challenges for end-to-end generation. In Kristiina Jokinen, Manfred Stede, David DeVault, and Annie Louis, editors, *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206, Saarbrücken, Germany, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-5525. URL https://aclanthology.org/W17-5525/. 2