Finetuning GPT-2 on E2E Using LoRA

Ryan Swift
Thomas Lord Department of Computer Science
University of Southern California
Los Angeles, CA
ryanswif@usc.edu

Abstract

We created our own implementation of Low-Rank Adaptation (LoRA) and used the original configuration for finetuning GPT-2 Medium on the E2E NLG Challenge dataset provided by [3] to establish a baseline. We then developed an improvement upon that baseline, making use of the high degree of control over complexity and performance offered by the LoRA framework. Our manually-implemented improved configuration outperformed the baseline results provided for the E2E dataset by an average of 4.74%, improved upon the baseline configuration with GPT-2 Small by an average of 6.85%, and achieved $\pm 1\%$ of the performance of our goal on each of BLEU, NIST, METEOR, ROUGE_L, and CIDEr.

1 Introduction

The baseline we chose to replicate for this project was the performance on the E2E NLG Challenge by GPT-2 Medium finetuned using Low-Rank Adaptation (LoRA), as reported in [3]. Performance on the E2E NLG Challenge is reported using five metrics: BLEU [6], NIST[1], METEOR [2], ROUGE_L [4], and CIDEr [8]. Due to resource constraints, we instead finetuned GPT-2 Small and adjusted our goal for the improvement to be scores equal to the official E2E NLG Challenge baseline[1] plus 80% of the delta from the official baseline to GPT-2 Medium finetuned with LoRA. All code for this project can be found at `https://github.com/ryanaswift7/gpt2_LoRA_567`.

1.1 E2E NLG Challenge

The E2E NLG Challenge is a benchmark task in the field of Natural Language Generation (NLG). Its goal is to generate natural language descriptions from tabulated meaning representations (MRs) specialized for the restaurant and dining domain. Each MR consists of 3–8 attributes (slots), such as name, food or area, and their values. The challenges of the E2E dataset lie in its size, lexical richness, syntactic variation and discourse phenomena, and content selection [5]. There is an official script for reporting BLUE, NIST, METEOR, ROUGE_L, and CIDEr scores which can be found at `https://github.com/tuetschek/e2e-metrics`.

1.2 Low-Rank Adaptation (LoRA)

LoRA is a method of parameter-efficient finetuning introduced in [3]. Large language models (LLMs) have an enormous number of parameters, making it extremely inefficient to re-train the entire model when finetuning for a specific application. Instead of training the original model, LoRA instead freezes all of the original parameters $W_{original}$ and injects and trains two low-rank matrices (A and B) into a given layer. The output of the layer becomes

[1] baseline results achieved using TGen [2], [5]

Preprint. Under review.

$$W_{LoRA} = W_{original} + \Delta W,$$

where $\Delta W = BA$ is the transformation learned during the LoRA finetuning process. Thus, the challenge of finetuning a model with potentially billions of parameters is reduced to training a number of far smaller matrices to learn how to adapt the original outputs of the model into outputs suitable for the chosen task.

2 Methods

We implemented LoRA from scratch and injected LoRA modules into GPT-2 Small, which we then finetuned on the E2E NLG Challenge dataset. To establish a baseline, we took the same LoRA configuration used in [3] to finetune GPT-2 Medium on E2E and applied it to GPT-2 Small. To improve upon this baseline, we created a new LoRA configuration with changes intended to make the resulting model more performant while also increasing the number of trainable parameters. Some of the notable changes made in the improved configuration include increasing the rank of the LoRA matrices and injecting LoRA modules into more layers of GPT-2 Small than in the baseline configuration. Both the baseline and improved models were finetuned with identical training parameters as the original implementation on GPT-2 Medium in [3].

3 Implementation Details

All of the code for this project was written in Python, as it is the language most widely used for machine learning tasks and research. Everything except the training loop and metric computation was manually implemented for this project. For the training loop, we used HuggingFace's Trainer to hold the LoRA models and training arguments and used it's *train()* method to dispatch the model training. Given that the E2E NLG Challenge provides an official script to extract metrics from text generated from the *test* dataset, that script was used to compute the results.

3.1 Manual Implementation

There were four categories of code which were implemented manually for this project: utilities, data processing, LoRA modules, and LoRA models.

3.1.1 Utilities

There were two utilities created, each solving a problem for experimental setup. The first enforced reproducibility of the project by setting a fixed seed across all relevant Python modules, as well as forcing CUDA to behave deterministically. The second simply handled the case of a multi-GPU environment by telling CUDA to use only one of the available devices.

3.1.2 Data Processing

Data processing involved preprocessing the input data, adjusting the tokenizer in the LoRA model to understand the inputs, generating test outputs, and postprocessing of outputs to create appropriately-formatted files for the official E2E metric script.

Preprocessing Though the E2E dataset was easy to import using HuggingFace's *datasets* library, the data still require preprocessing to be comprehensible to GPT-2. We used a lambda function to map each combination of meaning representation (*MR*) and human reference (*ref*) into a single string contaning the *MR* and *ref*.

Tokenizer The tokenizer ended up being a significant source of the issues encountered during the course of the project's development. Naturally, the model needed to have some indicator if where the *MR* ended and the *ref* began. Initially, we used the end-of-sequence token as an indicator, and inserted it between the *MR* and *ref*. This resulted in abysmal model performance and was promptly abandoned. The solution we eventually reached was creating a new *reference* attribute in the tokenizer and defining a special token for it.

Output Generation One of the quirks of text generation was that the input text is also included in the output. The natural solution to this was taking the generated outputs and splitting on the *reference* token. This required allowing the tokenizer to include special tokens in the output, which meant that all special tokens present needed to be filtered out.

Postprocessing Since the official metric script expects inputs to follow a standard of formatting, the reference file for the *test* data had to be carefully generated such that all human references associated with the same meaning representation were grouped together.

3.1.3 LoRA Modules

LoRA modules were where the A and B matrices were created and initialized, as well as where the forward pass was implemented. Two versions of initialization were implemented: the original from [3], and Gaussian initialization.

- **Original**: A is initialized with random Gaussian values and scaled by the reciprocal of the rank. B is initialized to zeroes. Thus, $BA = 0$ at the start of finetuning.
- **Gaussian**: Similar to the original, except B is initialized in the same manner as A.

3.1.4 LoRA Models

The Python class created for the LoRA models accounted for the bulk of the implementational complexity of the project. It freezes the original model's parameters, identifies which layers will be given LoRA modules, injects the modules alongside the model's original layers, and updates the forward method for each of the LoRA-modified layers to pass the input through both the original path and through the LoRA transformation in parallel, then sum the results.

4 Experiments

4.1 Results

We consider two different baselines for our project: the official baseline for the E2E NLG Challenge, and the performance of GPT-2 Small using the original LoRA configuration for finetuning GPT-2 Medium on E2E from [3]. The baseline and improved LoRA configurations ae listed in Table 1. The training and inference parameters are taken directly from [3] and listed in Table 2. We present the scores for both baselines, the original LoRA implementation, our goal, and our improved model in Table 3. A comparative analysis of the results of our improved model versus the other four models is presented in Table 4.

Table 1: LoRA configurations

	Baseline	Improved
Rank	4	16
α	32	32
Target Modules	*c_attn*	*c_attn, c_proj, c_fc*
Dropout	0.1	0.1
Bias	none	none
Initialization	Original	Gaussian

Table 2: Hyperparameters

Training	
Optimizer	AdamW
Weight Decay	0.01
Batch Size	8
# Epoch	5
Warmup Steps	500
Learning Rate Schedule	Linear
Label Smoothing	0.1
Learning Rate	0.0002
Inference	
Beam Size	10
Length Penalty	0.9
no repeat ngram size	4

Table 3: Performance metrics

Model	# Trainable Parameters	BLEU	NIST	METEOR	ROUGE_L	CIDEr
Baseline (E2E)	Not Provided	0.6593	8.6094	0.4483	0.6850	2.2338
Baseline (GPT-2 S)	0.15M	0.6476	8.4059	0.4381	0.6588	2.2413
Original (GPT-2 M)	0.35M	0.704	8.85	0.468	0.7180	2.53
Goal	N/A	0.695	8.80	0.464	0.711	2.47
Improved	2.36M	0.6882	8.7540	0.4667	0.7103	2.4537

Table 4: Relative improvement of improved LoRA model

Model	BLEU	NIST	METEOR	ROUGE_L	CIDEr	Average
Baseline (E2E)	+4.38%	+1.68%	+4.10%	+3.69%	+9.84%	+4.74%
Baseline (GPT-2 S)	+6.27%	+4.14%	+6.53%	+7.82%	+9.48%	+6.85%
Original (GPT-2 M)	-2.24%	-1.08%	-0.278%	-1.07%	-3.02%	-1.54%
Goal	-0.993%	-0.523%	+0.569%	-0.098%	-0.660%	-0.341%

4.2 Computational Resources

We trained our models on two NVIDIA RTX A6000's connected though NVLink, for a total of 96 GB of VRAM.[2] The entire pipeline of model creation, data download and processing, model training, and output generation took about 1 hour, with each model taking no more than 4 GB to train.

5 Conclusion

We were successfully able to create and original implementation of LoRA and use it to finetune an LLM for table-to-text generation. After establishing a baseline with GPT-2 Small, our improved configuration produced a model which outperformed both our baseline and the baseline results reported for the dataset it was finetuned on. Additionally, we achieved near identical scores to our goal, indicating that our improved model achieved roughly 80% of the improvement that the original LoRA implementation with GPT-2 Medium achieved over the E2E baseline. Despite GPT-2 Small requiring 2.35M LoRA parameters to achieve only 80% of the performance that GPT-2 Medium achieved with just 0.35M, the additional 2M LoRA parameters amounts to less than 1% of the

[2]We did not have access to these resources at the time of the proposal, hence the decision to train GPT-2 Small instead of Medium

parameter difference between the Small and Medium versions of GPT-2 [7]. It is thus abundantly clear that LoRA's combination of efficiency and performance make it an excellent finetuning framework.

References

[1] George Doddington. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*, pages 138–145, 2002.

[2] Ondřej Dušek and Filip Jurčíček. A context-aware natural language generator for dialogue systems. In Raquel Fernandez, Wolfgang Minker, Giuseppe Carenini, Ryuichiro Higashinaka, Ron Artstein, and Alesia Gainer, editors, *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 185–190, Los Angeles, September 2016. Association for Computational Linguistics.

[3] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

[4] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[5] Jekaterina Novikova, Ondrej Dušek, and Verena Rieser. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, Saarbrücken, Germany, 2017. arXiv:1706.09254.

[6] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In Pierre Isabelle, Eugene Charniak, and Dekang Lin, editors, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[7] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[8] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575, 2015.