Train Your Own Language Models

1 Project Introduction

The purpose of this project is to train a GPT model. The basic requirements include:

1. (Language modeling) Pretrain the GPT model.
2. (Instruction tuning) Finetune the model with a instruction-following dataset to align the model with human.
3. Submit your code and a technical report including method, experimental setup and results. You can use the NeurIPS template[1] for your report.

This project aims to let students understand the entire pipeline of training a language model. Given the limited resources, we cannot train large models with limited memory and limited time. Therefore, we will not mainly evaluate your project based on quantitative results, but instead evaluate the completeness and richness of your methods and experiments. The technical report should include at least:

1. Pretraining approach. Describe your pretraining data, model architecture, training details, training cost, training loss curves, etc. Describe any extra technique if you use in pretraining.
2. Finetuning approach. Describe your finetuning method, data and training details.
3. Evaluation. Report your evaluation results from multiple perspectives.

You can refer to the Llama and Llama2 paper [7, 8] for the organization and content of your report.

2 Details

2.1 Model Architecture

We need to use a GPT-type model for the project. GPT (Generative Pretrained Transformer) is a transformer-based model with causal attention. It can be trained on unlabelled datasets, with the training objective to predict the next token, which also makes it known as an autoregressive model. For our project, we recommend that you try a model of around 1M parameters. For the model code, you can learn the model architecture from miniGPT[2] and write the code with HuggingFace[3]. Note that there are many variants of GPT with tiny technical changes. For released trained model weights, refer to the website[4].

Given limited resources, you can make a trade-off between the model size and training time by following empirical neural scaling laws [4, 3]. Also, you are encouraged to use advanced improvements on your model. See section 2.2 in Llama [7] for reference.

[1] `https://neurips.cc/Conferences/2023/PaperInformation/StyleFiles`
[2] `https://github.com/karpathy/minGPT`
[3] `https://github.com/huggingface/transformers`
[4] `https://huggingface.co/roneneldan/TinyStories-1M`

2.2 Pretraining Data

Pretraining data is usually large-scale unlabelled text data, such as web pages, news, books, etc. They often include a lot of knowledge, in order that the model can learn abilities in various aspects. However, the quality of these datasets varies greatly. We cannot guarantee that every document in the datasets is grammatically and semantically correct, nor can we ensure that they are all worthwhile for the model to learn from. On the other hand, the capacity of small models makes it challenging for them to learn such a large amount of knowledge. Therefore, we recommend using the dataset[5]. More contents can be found in [1]. Pretraining on this dataset will enable (small) models to generate English stories well. You are encouraged to use more challenging but practical datasets. See section 2.1 in Llama [7] for reference.

2.3 Instruction Tuning Data

Why should we do instruction tuning? Instruction tuning gives the model the ability to perform specific tasks. Although pretraining gives the model basic language understanding and other general capabilities, it does not have enough ability to perform specific tasks. By finetuning the model with instructions, we can enable the model to perform the task of engaging in chat conversations with humans.

Instruction tuning data is usually labelled data with clear task instructions, such as a set of question and answer pairs, dialogues, etc. Please note that even though the data is labeled, the training objective remains unchanged during the supervised finetuning stage. For example, instruction tuning data can refer to Alpaca[6]. In addition to supervised finetuning, current researchers mainly use RLHF methods [6] to achieve better alignment results. Although the quality of the conversation on the small model is hard to guarantee, you are encouraged to use this better alignment method.

2.4 Evalution

The model can be evaluated from multiple perspectives. For example, for the evaluation of the language modeling task itself, you can use the Wikitext dataset [5]. For reasoning capabilities, you can use the MMLU dataset [2]. For the quality of language generation and conversational abilities, you can use qualitative methods to show some of the output content of your trained model, demonstrating that your model has certain language generation capabilities. You must evaluate your model using qualitative and one quantitative method. And you are encouraged to evaluate the model from multiple perspectives. See section 3 in Llama [7] and section 2.3 in Llama2 [8] for reference.

3 Remarks

We recommend you to use the HuggingFace[7] to complete the entire project. For instance, you can refer to their GPT-2 training script example[8]. They also provide some detailed tutorials, please refer to the tutorial[9].

This project has a certain degree of freedom, you are encouraged to design interesting experiments to explore and report the results.

References

[1] Ronen Eldan and Yuanzhi Li. Tinystories: How small can language models be and still speak coherent english? *arXiv preprint arXiv:2305.07759*, 2023.

[5] `https://huggingface.co/datasets/roneneldan/TinyStories`

[6] `https://github.com/tatsu-lab/stanford_alpaca`

[7] `https://github.com/huggingface/transformers`

[8] `https://github.com/huggingface/transformers/tree/main/examples/pytorch/language-modeling`

[9] `https://huggingface.co/learn/nlp-course/chapter7/6?fw=pt`

[2] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

[3] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[4] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[5] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

[6] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

[7] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[8] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.