Umair Imran1

The Complete Guide for Public Models

Quick Submit
Quick Submit
National University of Computer and Emerging Sciences, Islamabad

Document Details

Submission ID
trn:oid:::1:3131414113

Submission Date
Jan 17, 2025, 3:11 PM GMT+5

Download Date
Jan 17, 2025, 3:17 PM GMT+5

File Name
The_Complete_Guide_for_Public_Models.pdf

File Size
310.8 KB

8 Pages

1,198 Words

6,905 Characters

0% detected as AI

The percentage indicates the combined amount of likely AI-generated text as well as likely AI-generated text that was also likely AI-paraphrased.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Detection Groups

1 AI-generated only 0%
Likely AI-generated text from a large-language model.

2 AI-generated text that was AI-paraphrased 0%
Likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?
The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?
Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.

The Complete Guide for Public Models & Exploring Methods for Fine Tuning

Muhammad Umair Imran

umairimran627@gmail.com

National University of Computer and Emerging Sciences

January 17, 2025

Contents

Abstract

This report aims to provide clear guidelines for the selection of publicly available open-source models. It also explores various methods for fine-tuning these models depending on specific problem cases, resource availability, and performance requirements.

Open Source Models

Publicly available open-source models can be utilized for a variety of purposes. Below are some notable models along with their specific use cases and details:

- **Llama 3.1 8B Instruct**: A model specifically trained for instructions, such as conversations.
  `https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct`
- **Mistral 7B Instruct**: Trained for instruction-following tasks.
  `https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2`
- **bigscience/bloomz-560m**: Best for language translation with support for over 46 languages.
  `https://huggingface.co/bigscience/bloomz-560m`
- **google/gemma-2-2b-it**: A model well-suited for small-resource environments such as running on a laptop or edge devices.
  `https://huggingface.co/google/gemma-2-2b-it`
- **tiiuae/falcon-180B-chat**: Optimized for large-scale conversational applications.
  `https://huggingface.co/tiiuae/falcon-180B-chat`
- **Salesforce/xgen-7b-8k-inst**: Ideal for large context windows, especially for business-specific needs.
  `https://huggingface.co/Salesforce/xgen-7b-8k-inst`
- **01-ai/Yi-1.5-34B-Chat**: Enhances chat performance in both English and Chinese.
  `https://huggingface.co/01-ai/Yi-1.5-34B-Chat`

- **openai-community/gpt2**: A general-purpose model but requires fine-tuning with specific instructions for optimal results.
  `https://huggingface.co/openai-community/gpt2`

Fine Tuning Methods

PEFT: Parameter Efficient Fine-Tuning

Parameter Efficient Fine-Tuning (PEFT) refers to methods that focus on efficiently adapting large pre-trained models for specific downstream tasks without requiring full model retraining. PEFT techniques aim to save computational resources by reducing the number of trainable parameters.

Low-Rank Adaptation (LoRA) of Language Models

Low-Rank Adaptation, or LoRA, involves freezing the pretrained model weights while introducing trainable rank decomposition matrices into each layer of the Transformer architecture. This approach significantly reduces the number of trainable parameters for downstream tasks, making it a parameter-efficient method for fine-tuning large language models.

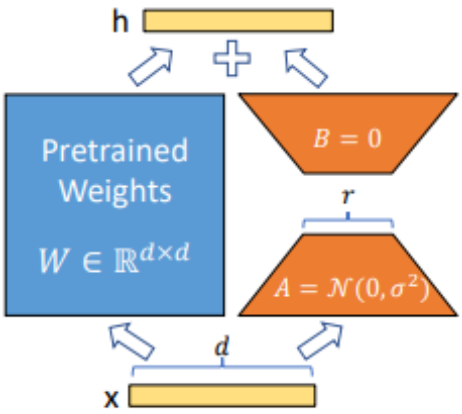

Figure 1: Low-Rank Adaptation Architecture.

LoRA works by converting high-rank matrices into low-rank matrices, which are then

combined and added back into the original matrices. This allowed model to have its original speciality while being modified for specific problem. As a result, lora improves the training efficiency without changing the model over all performance. [1].

QLoRA: Quantized Low-Rank Adaptation

In a large language model, parameters have to be efficient because that make them more suitable for deployment in low or limited resources.If we take a model, such as a 16 bit llama with 65 Billion parameters, that would require about 780 Giga Bytes of GPU memory, it can become expensive to use in real world apps.One Method to solve this problem is by using quantized model. The Process involves decreasing the model's accuracy, usually from 16 bit to 4 bit or 8 bit.

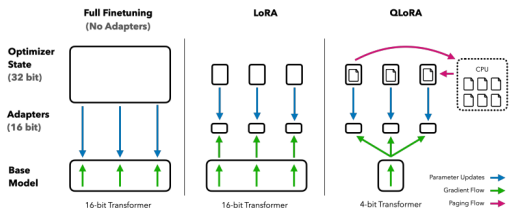

Figure 2: QLoRA

When the quantization of model is done, it is then further fine tuned through LoRA. By compromising on precision and then applying Lora, the model's parameters can be fine tuned for specific tasks, leading to major resource and time cost, while keeping most of the original model's performance properties.

For more details, you may look at this paper **QLoRA**, as explained by Dettmers et al. [2].

Prefix Fine Tuning

By Using This method, weights of model are not changed ; instead, we just add a prefix during fine tuning process. For example, if we are querying, "What is the president of Pakistan?", we simply add a prefix, such as [history-related], to save time and computational resources. After fine-tuning, the model has already been adapted, so the prefix does not need to be added during inference.

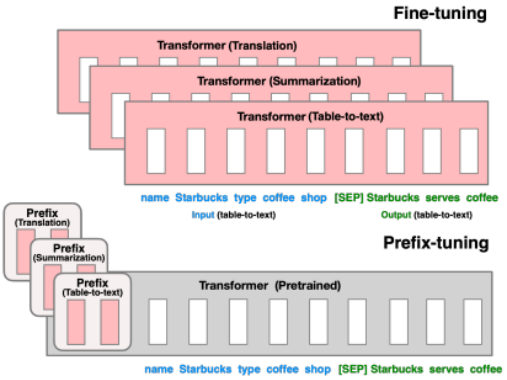

Figure 3: Prefix Fine Tuning

For more details on prefix tuning, see Liang and Liang [3].

Adapters

Instead of performing full fine-tuning, adapter layers are added between the model's forward layers. Full fine-tuning adjusts all parameters, requiring significant resources. However, in the this approach ,we add task specific layers to the model, making it capable for more use cases.This method saves resources, but adding additional layers can make inference slower due to sequential processing but that is bearble if we are not in critical areas such as medical.

For further details on adapters, you may refer to Hu et al. [4].

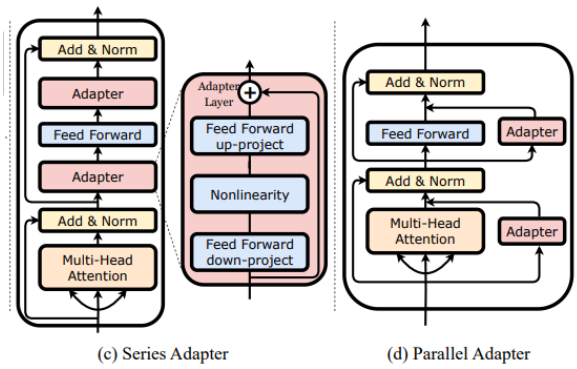

Figure 4: Adapters

Prompt Fine Tuning

In this prompt fine tuning, the model is fine tuned by using human prompts or AI generated soft prompts during the fine tuning phase. This method is computationally low cost and can also be altered to specific tasks.

For instance, in a sentiment analysis task, a soft prompt could be structured like this:

- **Setup:** The input would be like this: [soft_prompt_embeddings] + "This movie review is like this:" + [review_text] - **during the process of training,** the model changes [soft_prompt_embeddings] to better understand the task. - **Inference Input:** [soft_prompt_embeddings] + "This movie review is:" + "An exciting Bike ride full of haunt" - **Expected Output:** "Positive"

Another option is a hard prompt is manually added to the input :

- **Input:** "sentiment classification : positive or negative . review: An exciting Bike ride full of haunt." - **Output:** "positive"

More explanation on this topic can be found at [5].

P-Tuning

This technique of P tuning involves adding computer generated embeddings with the task specific prompt for prediction. Let Consider an example , input text may be structured as: [P-tuning embeddings] + [original task input text].

This approach works well on large models but it is not able to produce optimal results on small models .

For more information on P-Tuning, see Liu et al. [6].

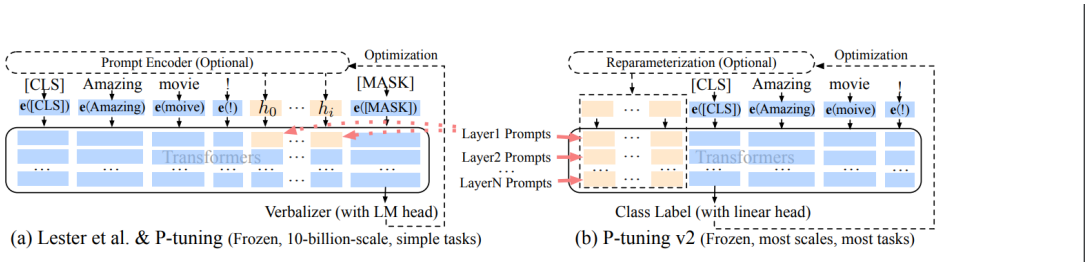

Figure 5: P-Tuning

References

[1] E. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-Rank Adaptation of Large Language Models," Microsoft Corporation, [Online]. Available: https://arxiv.org/abs/2106.09685v2. [Accessed: 7-Jan-2025].

[2] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient Finetuning of Quantized LLMs," arXiv, May 2023. [Online]. Available: https://arxiv.org/pdf/2305.14314.

[3] X. L. Li and P. Liang, "Prefix-Tuning: Optimizing Continuous Prompts for Generation," arXiv, 2021. [Online]. Available: https://arxiv.org/pdf/2101.00190. [Accessed: 07-Jan-2025].

[4] Z. Hu, L. Wang, Y. Lan, W. Xu, E.-P. Lim, L. Bing, X. Xu, S. Poria, and R. K.-W. Lee, "LLM-Adapters: An Adapter Family for Parameter-Efficient Fine-Tuning of Large Language Models," in Proc. of the 2023 Conf. on Empirical Methods in Natural Language Processing (EMNLP), Dec. 2023. [Online]. Available: https://aclanthology.org/2023.emnlp-main.319.pdf.

[5] S. Shah, "Prompt-Tuning: A Powerful Technique for Adapting LLMs to New Tasks," Medium, [Online]. Available: https://medium.com/@shahshreyansh20/prompt-tuning-a-powerful-technique-for-adapting-llms-to-new-tasks-6d6fd9b83557.

[6] X. Liu, K. Ji, Y. Fu, W. L. Tam, Z. Du, Z. Yang, and J. Tang, "P-Tuning v2: Prompt Tuning Can Be Comparable to Fine-Tuning Universally Across Scales and Tasks," arXiv preprint arXiv:2110.07602. [Online]. Available: https://arxiv.org/pdf/2110.07602.

[7] $HuggingFace, "PEFTPackageReference, "[Online].Available : https : //huggingface.co/docs/peft/en/package_reference/p_tuning.$