Mini LIMA

Binglin Zhou
521030910361

Quanquan Peng
521020910182

Yuhao Wang
521030910330

Abstract

Large language models, such as the Qwen, are typically trained through a dual-phase process: unsupervised pretraining from extensive raw text datasets and supervised fine-tuning (SFT) tailored to specific tasks. In this mini-paper, we explore the efficacy of “less is more for alignment” (LIMA) by deploying minimal instruction-based fine-tuning. We employed approximately 170 seed tasks to generate around 1,500 instructional outputs aimed at training these models. We compare two instructional generation methodologies and their impacts on the model’s performance in SFT. Our findings reveal that text length within the instructional outputs influences performance: shorter texts generally degrade model effectiveness, whereas longer texts enhance it, underscoring the importance of optimal text length in model training.

1 Introduction

The advent of large language models (LLMs) has revolutionized numerous applications in natural language processing, offering unprecedented capabilities in text generation, comprehension, and interaction. Typically, the development of these models involves a two-stage training process: unsupervised pretraining on vast corpora of raw text, followed by supervised fine-tuning (SFT) or reinforcement learning to tailor the models to specific tasks and user preferences. While this methodology has proven effective, it often requires substantial computational resources and extensively curated training datasets.

Recent approaches, such as the "less is more for alignment" (LIMA) strategy (Zhou et al., 2024), suggest that high-quality model outputs can be achieved with significantly fewer training examples, focusing on the quality rather than the quantity of training data. This approach challenges the conventional wisdom that extensive fine-tuning is necessary for model alignment and task performance.

In this study, we explore the practicality and effectiveness of applying a LIMA-like strategy to the Qwen1.5-1.8B and 0.5B models (Bai et al., 2023). Using a curated set of approximately 170 seed tasks, we generated around 1,500 instructional outputs to train these models. We aimed to investigate how different instructional generation methodologies impact the models’ ability to align with specific tasks during SFT. This research contributes to a better understanding of the efficiency of training strategies in large-scale model applications: our findings suggest that text length significantly affects the model’s performance. Specifically, shorter text lengths were found to adversely impact the model’s ability to perform well, whereas appropriately increasing the length of the instructional text enhanced the model’s performance.

2 Data Construction

The initial dataset provided for this work consists of 174 distinct examples. Each example is composed of several integral components: a task name, a set of instructions, a single input-output instance, and something not important, collectively designed to serve as the seed tasks for training and evaluation in our experiments. To augment and diversify this dataset, we incorporated principles from the self-instruct method (Wang et al., 2023). This methodology advocates for a dynamic and iterative approach to data augmentation, leveraging the original dataset to generate new examples through techniques such as paraphrasing, expanding, and refining existing data. This strategy not only increases the volume of data available for training but also enhances the variety and complexity of the tasks, thereby providing a richer foundation for the development of more sophisticated and adaptive machine learning models.

```
Following are some examples of {name, instruction}
pairs. Please generate more pairs like these.
Maintain the same format and be as creative as
possible.
```

Prompt 1

```
Following is a JSON line with name, instruction,
and instances. Given some instances, please
generate two more instances based on the provided
information, add them to the JSONL file, and
return it in the same line format.
```

Prompt 2

```
Following data is in a JSON line with name and
instruction. Please generate five instances
according to the example data, and return them in
the format of the example data in a JSONL format.
Ensure that the name, instruction, and instances
conform to the example data's format.
```

Prompt 3

```
Given is a question-answer task with task name,
instruction and input-output instance. The output
text is too short, insipid, dull, boring. Let it
be more creative, detailed, interesting, engaging,
fascinating, and specific. Generate more creative
words for the output part according to the task
name, instruction and input. Here is a example:
Data give to you: Task name: '{ex_name}',
Instruction: '{ex_instr}', Input: '{ex_input},
output: '{ex_output}'. Output should be:
{ex_expanded}. Example part end. Following is
content of the task: Task name: '{name}',
Instruction: '{instr}', Input: '{input}, output:
'{output}'. Print output text directly.
```

Prompt 4

Figure 1: The prompt text described above is utilized to invoke the *GPT-3.5 turbo* API, with its application delineated as follows. The initial three prompts are employed in the naive method, while the final prompt is applied in the improved method.

2.1 Naive Method

To embody the principle that *less is more*, we adopted a straightforward coding strategy to expand our dataset. Initially, we doubled the number of tasks by leveraging all provided task names and instructions. We facilitated this expansion through a carefully crafted prompt as *prompt 1*.

Subsequently, for the original 174 tasks, each already equipped with a unique instance, we used a targeted prompt to generate additional instances. This prompt specified as *prompt 2*.

For newly generated tasks, we implemented a one-shot learning approach using data from previous tasks to instruct the model. The directive for the *GPT-3.5 turbo* model was described as *prompt 3*.

This methodical approach streamlined the data expansion process.However, it became evident that this initial method was overly naive, and the outcomes were suboptimal during subsequent training and evaluation phases. Consequently, we implemented several refinements to enhance the method's effectiveness and improve the quality of the generated data.

2.2 Improved Method

To determine the underlying issues in our initial approach, it is essential to conduct a detailed analysis of the subsequent experiments. This analytical process will enable us to pinpoint the shortcomings and guide the implementation of the following enhancements.

Following thorough discussion and verification, we identified that a significant factor contributing to the low-performance scores in our experiments was the inadequate length of the output text. To rectify this issue, we employed a specific API designed for text extension, as detailed in *prompt 4*.

3 Training

After constructing a small, high-quality dataset, we leveraged it to **S**upervise **F**ine **T**une (**SFT**) two open-source Large Language Models. For this project, we selected Qwen1.5-0.5B and Qwen1.5-1.8B as the base models. To facilitate training, we employed DeepSpeed, which optimizes resource use on a single RTX 3090 GPU. We addressed GPU memory constraints by implementing gradient accumulation and CPU Offload in the DeepSpeed ZeRo-2 stage configuration. The batch size for Qwen1.5-0.5B was set to eight, whereas for Qwen1.5-1.8B, it was set to two. We maintained consistency in other hyperparameters across both training setups. The common hyperparameters are set as follows:

optimizer: AdamW
learning rate: 1×10^{-5}
weight decay: 3×10^{-7}
betas: $(0.8, 0.999)$
scheduler: WarmupLR
warmup steps: 100

4 Evaluation

Here we use AlpacaEval (Dubois et al., 2024) to evaluate the model's performance and show the evaluation result:

To evaluate the impact of Supervised Fine-Tuning (SFT) on the Qwen model, we utilized the `gpt-3.5-turbo-0301` as an annotator. As shown in Table 1, this decline indicates that both two models' overall performance are compromised. Compared to the previous result, our improved method shows improvement in the Qwen-0.5B model in Tabel 2.

	Baseline	**SFT**
Qwen-1.8B	31.22	12.87
Qwen-0.5B	8.08	6.38

Table 1: Qwen's Win Rate on AlpacaEval w/ Naive Method

	Baseline	**SFT**
Qwen-1.8B	31.22	24.62
Qwen-0.5B	8.08	14.55

Table 2: Qwen's Win Rate on AlpacaEval w/ Improved Method

5 Conclusion

Throughout our investigation of the Qwen models utilizing the LIMA strategy, we have observed that the choice of prompts for seed tasks plays a crucial role in the model's performance post-SFT. Specifically, our results indicate that not only the relevance but also the specific construction of prompts markedly affects the alignment and effectiveness of the model in executing assigned tasks. Additionally, we discovered that the length of the instructions has a impact on model performance, with shorter instructions generally leading to poorer outcomes after fine-tuning.

6 Limitations

In our experiments, we observed an intriguing phenomenon: for the Qwen-1.8B model, performance decreased under the improved method in Table 2. This issue is worthy of discussion as the 0.5B model's performance increases; however, we do not intend to conduct a deeper investigation into this matter here due to time constraints.

Acknowledgments

We thank Prof. Pengfei Liu for his guidance and support throughout this project. We also thank all the TAs for their dedicated assistance.

References

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. 2024. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. *Preprint*, arXiv:2212.10560.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2024. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36.