Domain-Specific Fine-Tuning of Large Language Models for Interactive Robot Programming

Benjamin Alt[1], Urs Keßner[1,2], Aleksandar Taranovic[2], Darko Katic[1], Andreas Hermann[1], Rainer Jäkel[1], and Gerhard Neumann[2]

[1] ArtiMinds Robotics, 76131 Karlsruhe, Germany,
benjamin.alt@artiminds.com
[2] Karlsruhe Institute of Technology, Autonomous Learning Robots Lab, 76131 Karlsruhe, Germany

Abstract. Industrial robots are applied in a widening range of industries, but robot programming mostly remains a task limited to programming experts. We propose a natural language-based assistant for programming of advanced, industrial robotic applications and investigate strategies for domain-specific fine-tuning of foundation models with limited data and compute.

Keywords: large language models, robot programming, fine-tuning

1 Introduction

Industrial robot programming is cumbersome and costly, especially for advanced applications requiring force sensing and vision. Skill-based approaches simplify programming complex robot tasks by sequencing learned or pre-programmed sub-skills such as grasping or peg-in-hole fitting. However, skill-based programming this still requires expert knowledge, e.g., physical dependencies between acceleration and contact forces, but also technical aspects, such as the applicability of skills in collision-prone or low-tolerance environments. With the introduction of large-scale pretrained foundation models such as LLaMA [6], there is an emerging body of work on fine-tuning large language models (LLMs) for use in various application domains [1, 9]. We propose a language-based programming assistant, which offers interactive dialogues about skills, example use-cases and expected robot behavior. We present three alternative model families, trained by three different domain-specific fine-tuning approaches. The variants are evaluated by comparing their BERTScore [8] performance as well as a user survey with industry experts.

2 Methods

As a testbed, we use the ArtiMinds Robot Programming Suite (RPS), an integrated development environment (IDE) for industrial robots. With it, robots are programmed using parameterizable robot skills (called "templates") such

as "Grasp", "Insert", etc. The resulting robot program is a tree of parameterized templates, which is compiled into executable code. The assistant answers questions of three types:

Providing high-level explanations of templates. Given a question like "What does a 'Move to State' template do?", it should answer something like "It moves the end-effector on a collision-free trajectory to a goal specified in configuration space." The complexity of answers range from simple, as illustrated here, to highly complex, particularly for force-controlled skills whose behavior depends in part on interactions with the environment.

Providing examples for the usage of templates. To a question like "When should I use a 'Move to State' template?", it should answer "A 'Move to State' template can be used to efficiently move the robot arm through environments in which collisions can occur." Examples can involve concrete application domains, such as painting, gluing or welding for a "Path Loader" template.

Providing step-by-step explanations of expected robot behavior. To a question like "What motions will a robot make when executing a 'Grasp' template?", the assistant should answer "It will execute a collision-free approach motion, open the gripper, move closer to the object, close the gripper, and depart with a collision-free motion." Step-by-step explanations provide useful information for very high-level templates such as "Peg in Hole".

We investigate how LLMs can be fine-tuned under the compute and data constraints typical for industrial small- or medium-sized enterprises (SMEs). This implies limitations to a single server-grade graphics processing unit (GPU), e.g., Nvidia A100, with 80GB VRAM. Traditional fine-tuning requires up to 780GB given 65B parameters [2]. To reduce model size, we leverage QLoRA adaptation with 4-bit NormalFloat quantization and double quantization [2]. We investigate three data sparse training regimes: Fine-tuning an instruction-following model on domain-specific data, which has been pretrained in different domains; fine-tuning a streaming model for instruction-following on domain-specific data; and fine-tuning a streaming model on a domain-specific streaming data, succeeded by domain-specific instruction-following training. Comparing these training regimes determines whether the use of general-purpose pre-trained models provides useful priors to simplify fine-tuning. Moreover, we investigate the impact of prefix fine-tuning [3] on model quality.

2.1 Datasets

Streaming dataset: One of the investigated training regimes involves the fine-tuning of a general-purpose streaming model such as LLaMA on domain-specific data. We create a dataset $\mathcal{D}_{stream}$ by parsing the ArtiMinds RPS user manual into a plain-text representation. Non-informational elements like the cover page, table of contents, index pages, and copyright paragraphs were removed to focus on the technical content.

Instruction-following datasets: We constructed a domain-specific dataset to evaluate whether a model could be trained to answer questions using only natural language instructions. The dataset contains questions about the usage of

ArtiMinds RPS templates, asking for template descriptions, use cases, and step-by-step descriptions extracted from the RPS documentation. Zhang and Soh [7] found that models may be overly sensitive to minor differences in input prompts, resulting in unintended variability in the generated responses. To address this, we generated 10 question variants for each information topic using ChatGPT 3.5 [4]. The resulting dataset $\mathcal{D}_{instr}$ contains a total of 250 instruction-label pairings.

Prefix fine-tuning is a technique in which a specific prefix is added to each prompt to guide the model into the correct domain [3]. To determine how this affects domain-specific fine-tuning, we created an additional version $\mathcal{D}_{instr}^{prefix}$ of the instruction-following dataset, with a custom, domain-specific prefixes, while $\mathcal{D}_{instr}$ applies the default Alpaca prefix [5].

2.2 Model Variants

Alpaca Finetuned on Instruction Task: We finetune three pretrained Alpaca models on our domain-specific instruction-following dataset $\mathcal{D}_{instr}$. The 7B and 13B models were fully fine-tuned (see Taori et al.), while the 30B model was finetuned using QLoRA, and the resulting LoRA adapters merged back into the original foundation model. Double 4-bit NormalFloat quantization [2] is applied. We perform QLoRA with rank $r = 8$, $\alpha = 16$ and dropout $= 0.05$ for all 4 self-attention matrices.

LLaMA Finetuned on Instruction Task: Instead of fine-tuning a pretrained instruction-following model, domain adaptation could also be achieved by fine-tuning a streaming model on an instruction-following dataset in the new domain. We use LLaMA models (7B, 13B and 30B) and quantize them to double-quantized 4-bit NormalFloat. We train a LoRA adapter on the instruction-following task $\mathcal{D}_{instr}$, using the same hyperparameters as in Section 2.2.

Merged LLaMA (Streaming & Instruction Task): A third variant of domain adaptation is to perform domain-specific fine-tuning in two stages: First on a streaming task, and then on an instruction-following task, both in the new domain. Using the same pretrained and quantized LLaMA models as in Section 2.2, we first train a LoRA adapter on the streaming dataset $\mathcal{D}_{stream}$. Then, we train an additional LoRA adapter on $\mathcal{D}_{instr}$.

Prefix Fine-tuning To assess to what extent prefix fine-tuning facilitates domain adaptation, we train additional variants of all models mentioned above on $\mathcal{D}_{stream}^{prefix}$ and $\mathcal{D}_{instr}^{prefix}$ respectively.

3 Experiments

3.1 BERTScore Evaluation

To assess the semantic correctness of the trained models' responses, we compute BERTScores [8] with an evaluation dataset. Nine instructions and reference responses to questions about RPS templates were manually written by domain experts. BERTScore recall, precision and F1 scores were calculated between each response and its reference.

Table 1. BERTScores and survey results for the 10 best-performing models, sorted by F_{BERT}. Factual correctness (C) and domain adherence (D) are binary features, perceived helpfulness (H) is ranked on a five-point Likert scale (1 - low, 5 - high).

	R_{BERT}	P_{BERT}	F_{BERT}	C	D	H	$N^{\dagger}$
Alpaca 30B	0.8156	0.7945	0.8042	0.3830	1.0000	3.0851	5
Alpaca 7B	0.8044	0.7979	0.8007	0.2041	0.8776	1.9796	5
Alpaca 13B w. p.*	0.7868	0.7929	0.7894	0.4000	0.8600	2.5000	5
Alpaca 7B w. p.	0.7873	0.7780	0.7818	0.2881	0.5085	2.4407	6
LLaMA 13B w. p.	0.8238	0.7421	0.7798	0.1800	0.7800	2.4600	5
Alpaca 30B w. p.	0.7722	0.7189	0.7438	0.0506	0.2785	1.3038	8
Merged LLaMA 7B w. p.	0.8017	0.6891	0.7386	0.2317	0.9146	2.2317	9
Merged LLaMA 30B	0.7952	0.6780	0.7311	0.1224	0.8776	2.0000	5
LLaMA 13B	0.7910	0.6747	0.7266	0.2143	0.6429	2.1714	7
LLaMA 30B	0.7631	0.6784	0.7174	0.1277	0.7872	1.8298	5

* with prefix † number of survey responses

The results are reported in Table 1. Alpaca models without prefix performed best with an F_{BERT} of 0.8. We observe that directly fine-tuning a pretrained instruction-following model yields the best results, and that model size seems uncorrelated to BERTScore performance.

3.2 User Survey

To obtain feedback about real-life performance, we conduct a survey with domain experts. The evaluation set contains 40 instructions, some of which outside the question types seen in training (e.g., "What is the difference between a 'Move to Point' template and a 'Move to State' template?"). A total of 33 engineers participated in the survey, each completing at least one questionnaire containing 10 randomly assigned prompts and model responses. Each prompt-response pair was evaluated on factual correctness ("The answer is precise and factually correct. (yes/no)"), domain adherence ("The answer remains in the domain of ArtiMinds RPS. (yes/no)") and perceived helpfulness ("Rate the helpfulness of the response on a scale from 1 to 5."). The results are shown in Table 1. Echoing the BERTScore results, Alpaca models performed best with respect to correctness, domain adherence and helpfulness. It must be noted, however, that all trained models have noticeable shortcomings with respect to correctness and overall helpfulness. The LLaMA-based models in particular struggled with sentence structure, often repeating words and predicting unknown tokens. In contrast, Alpaca-based models generated responses with satisfactory format. Prefix fine-tuning did not have a discernible impact on model performance.

4 Conclusions

We propose a natural-language-based programming assistant which offers human-like, interactive dialog for assistance with complex industrial robot programming. The assistant offers explanations of skills, example use-cases and expected robot behavior. We train three families of LLMs realizing three different approaches for domain-specific fine-tuning and evaluate them with respect to a quantitative semantic similarity metric as well as a user survey. The results indicate that domain-specific fine-tuning of instruction-following models achieves robust domain transfer. However, the results are not (yet) sufficient for practical use. The models often exhibit strong hallucinations (including out of domain responses) and repeat words or phrases or even switch from English to German. This suggests that fine-tuning alone is insufficient to imprint the new knowledge into the model with limited data. Prompting strategies, which eschew fine-tuning in favor of carefully designed prompts, are promising alternatives, and have become technically feasible with the most recent generation of foundation models. Likewise, larger-scale user surveys are required to draw more robust and nuanced conclusions about real-world model performance.

Acknowledgment

This work was supported by the German Federal Ministry of Education and Research (grant 01DR19001B).

References

1. Chalkidis, I., Fergadiotis, M., Malakasiotis, P., Aletras, N., Androutsopoulos, I.: LEGAL-BERT: The Muppets straight out of Law School (Oct 2020)
2. Dettmers, T., Pagnoni, A., Holtzman, A., Zettlemoyer, L.: QLoRA: Efficient Finetuning of Quantized LLMs (May 2023)
3. Li, X.L., Liang, P.: Prefix-Tuning: Optimizing Continuous Prompts for Generation. In: Proc. 59th Annu. Meet. Assoc. Comput. Linguist. 11th Int. Jt. Conf. Nat. Lang. Process. Vol. 1 Long Pap. pp. 4582–4597. Association for Computational Linguistics, Online (Aug 2021)
4. OpenAI: ChatGPT. https://chat.openai.com (Sep 2023)
5. Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., Hashimoto, T.B.: Alpaca: A Strong, Replicable Instruction-Following Model (Mar 2023)
6. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., Lample, G.: LLaMA: Open and Efficient Foundation Language Models (Feb 2023)
7. Zhang, B., Soh, H.: Large Language Models as Zero-Shot Human Models for Human-Robot Interaction (Mar 2023)
8. Zhang, T., Kishore, V., Wu, F., Weinberger, K.Q., Artzi, Y.: BERTScore: Evaluating Text Generation with BERT (Feb 2020)
9. Zheng, O., Abdel-Aty, M., Wang, D., Wang, C., Ding, S.: TrafficSafetyGPT: Tuning a Pre-trained Large Language Model to a Domain-Specific Expert in Transportation Safety (Jul 2023)