

ServiceTaskCoder: Building an AI-powered Chatbot for ServiceTask with Multi-Granular Instruction Tuning and AI-Guided Feedback

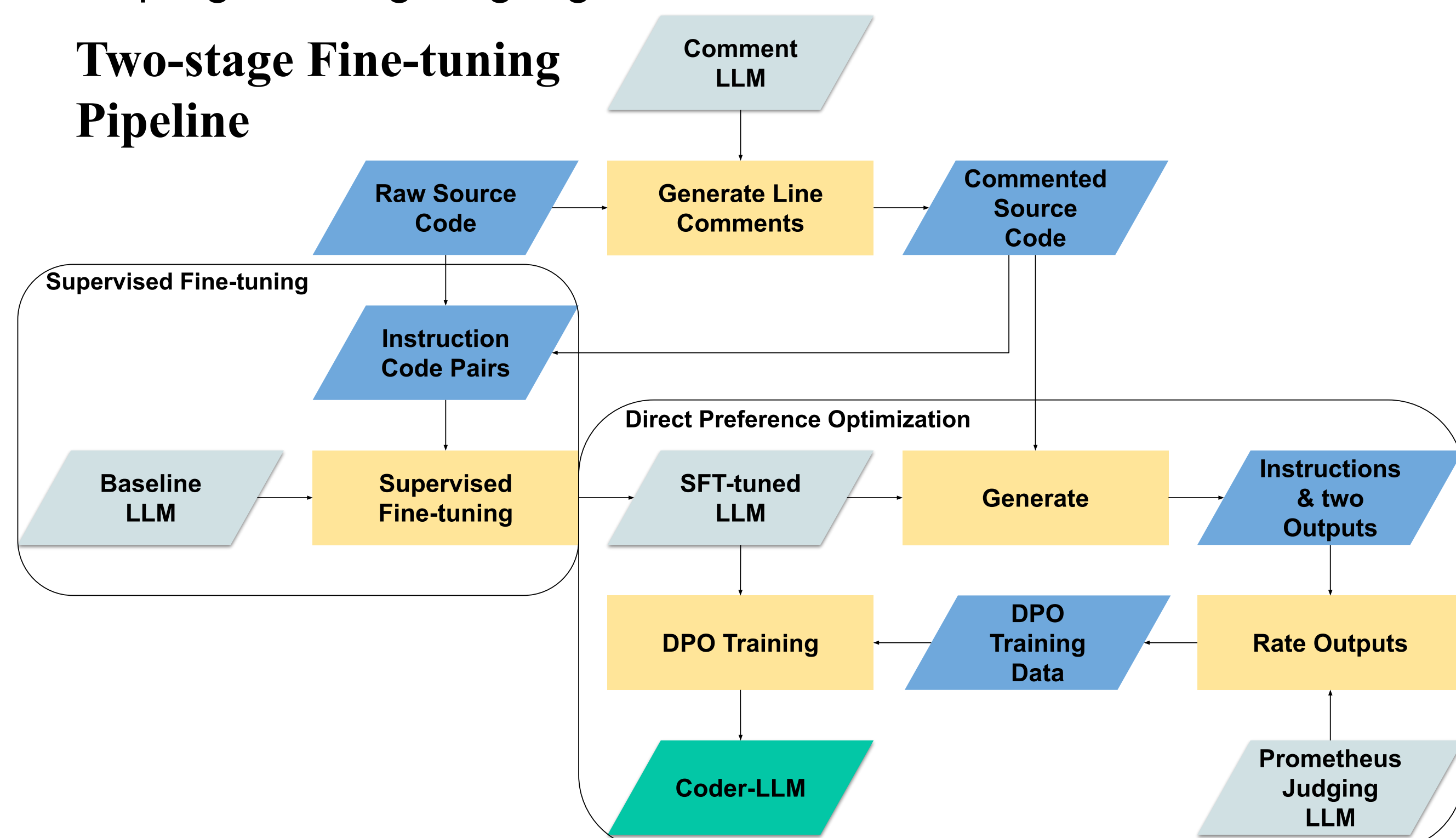


Vedad Misirlic*, Kevin Innerebner*, Gerald Stieglbauer, Elisabeth Lex

Abstract

Large Language Models (LLMs) have significantly advanced automatic code generation for popular general-purpose programming languages. However, their performance degrades on low-resource and 'proprietary' Domain Specific Languages (DSLs) such as *ServiceTask* due to scarce training data publicly available. We propose a fully automated pipeline to adapt coding LLMs with parameter-efficient Q-LoRA to *ServiceTask*, AVL's programming language. Our method consists of two stages: (1) we synthesize multi-grained instructions directly from raw code files by generating code-comment pairs. These instructions are used for Supervised Fine-Tuning (SFT); (2) we perform Reinforcement Learning with AI Feedback (RLAIF), where a commenting LLM produces entirely new, synthetic instructions, and our SFT model samples two outputs. A judging LLM selects a preference, and we subsequently optimize the model with Direct Preference Optimization (DPO). Results show that our method highly improves the quality of the code generated, and that the Chatbot improved syntactic and semantic understanding of the programming language.

Two-stage Fine-tuning Pipeline



Methodology

- Chat-based LLM based on Qwen2.5-7B-Coder-Instruct [1]
- Two-stage training process:
 - Supervised Fine-tuning (SFT)**
 - Generate comments for code blocks
 - Create instructions with comments and original *ServiceTask* code
 - Apply supervised fine-tuning using Q-LoRA
 - Direct Preference Optimization (DPO) with Reinforcement Learning with AI Feedback (RLAIF)**
 - Create new *ServiceTask* instructions with SFT model
 - Output candidate pairs
 - Prometheus v2 [2] judging LLM evaluates outputs based on grading rubrics
 - DPO with Prometheus choices applied to create the final model

How can we best adapt coding LLMs to *ServiceTask* with lightweight fine-tuning approaches where training material is limited?

Grading Rubrics [2]	Description of Rubric
Correct	Is the code that is generated correct, and does it work?
Formatted	Is the code that is generated well-formatted, concise, and readable?
Naming	Are the functions, methods, and variable names understandable and indicative?
Instruction-Following	Does the outputted code follow the original prompt which was defined?
Humble	Does the model's response avoid self-praise and apply self-critique?

Experiments

- Code Comment Quality**
 - Judging LLMs evaluate the code comments on a 1-7 Likert scale
- DPO and Grading Rubrics Positional Bias**
 - Positional bias investigation through choices swapping in grading rubrics prompt
- Proprietary Language Evaluation via Instruction-Based Code Generation**
 - Parsing success
 - Compiler warnings
 - chrF++ [3] similarity metric
- Proprietary Language Evaluation with LLM-as-a-judge**
 - Judging LLMs evaluate code quality of proprietary language
- Conventional Code Benchmarks**
 - HumanEval [4]
 - HumanEval+ [5]
 - Multipl-E [6]

Results

- Significant increase in output quality** in *ServiceTask*
- Approach **highly effective** in scenarios where few code examples and no tests are available for training
- No increase in code quality in high-resource programming languages such as Python

Conclusion

- We present a **lightweight solution applying a two-stage pipeline**: (1) Supervised fine-tuning through Instruction-code pairs, and (2) RLAIF and DPO for the final step
- Efficient approach** where data or computing power is **limited**

Sources

- [1] Hui, B., Yang, J., Cui, Z., Yang, J., Liu, D., Zhang, L., ... & Lin, J. (2024). Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.
- [2] Kim, S., Suk, J., Longpre, S., Lin, B. Y., Shin, J., Welleck, S., ... & Seo, M. (2024). Prometheus 2: An open source language model specialized in evaluating other language models. *arXiv preprint arXiv:2405.01535*.
- [3] Popović, M. (2017, September). chrF++: words helping character n-grams. In Proceedings of the second conference on machine translation (pp. 612-618).
- [4] Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. D. O., Kaplan, J., ... & Zaremba, W. (2021). Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.
- [5] Liu, J., Xia, C. S., Wang, Y., & Zhang, L. (2023). Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in Neural Information Processing Systems*, 36, 21558-21572.
- [6] Cassano, F., Gouwar, J., Nguyen, D., Nguyen, S., Phipps-Costin, L., Pinckney, D., ... & Jangda, A. (2023). Multipl-e: A scalable and polyglot approach to benchmarking neural code generation. *IEEE Transactions on Software Engineering*, 49(7), 3675-3691.

*These two authors contributed equally



<https://aisocietylab.github.io/>



TU Graz – Institute of Human-Centred Computing
Sandgasse 36/III, 8010 Graz, Austria, Tel.: +43 316 873-5624
office.hcc@tugraz.at, <https://hcc.tugraz.at/>

