Domain-Specific Language Models for Manufacturing Industry

Authors: *Hayato Uchide*, Tatsuhiko Saito*, Shingo Oidate**, Masahiro Hiramori**, Shinya Taguchi**

**AI Transformation Innovation Center, **Engineering and Development Center*

Abstract

With the advancement of Large Language Models (LLM), the practical use of Natural Language Processing (NLP) is progressing rapidly. Meanwhile, as models scale up, increases in computational cost and energy consumption, data privacy concerns, and constraints on real-time responsiveness among other issues are becoming apparent when operating them at manufacturing sites. In particular, in high-urgency situations directly tied to productivity—such as immediately identifying the cause of a production line stoppage and presenting response procedures—stable operation and high responsiveness of the model are required in restricted computing environments.

Against this backdrop, Mitsubishi Electric has developed a Small Language Model (SLM) specialized for the manufacturing domain, including the FA field. In this paper, in addition to continual pretraining and instruction tuning, we newly examined alignment capable of learning effectively even under limited data conditions. As a result, despite being a compact model with 1.8 billion parameters that can run on an edge device, it achieved an accuracy of 77.24% on a task that tests the correctness of knowledge in the FA field. This is anticipated to expand the scope of generative AI utilization in constrained environments such as manufacturing sites.

1. Introduction

With the advancement of LLMs, the practical use of NLP is progressing rapidly. In particular, OpenAI's ChatGPT[*1] and other general-purpose LLMs that run in the cloud deliver high performance across a wide range of language processing tasks, and are expected to be utilized in many industrial domains, including manufacturing. On the other hand, when using LLMs, the increase in computational cost and energy consumption that comes with larger models is becoming an issue.

Moreover, in operational settings, there are cases where simply using a general-purpose LLM in the cloud does not suffice. For example, at manufacturing sites, it is often difficult to send confidential information such as equipment manuals and incident response records outside of sites, and interactions with quick responses with on-site workers and equipment are frequently required. Specifically, these include immediately identifying the cause of a production line stoppage and presenting response procedures, providing support for confirmation during maintenance work, and making response decisions when abnormalities occur. In such cases, instantaneous information retrieval and responses based on confidential information such as equipment manuals and incident response records are directly tied to productivity and safety. In light of this backdrop, solutions that can be operated without relying on general-purpose LLMs in the cloud are needed. In particular, to complete secure, highly responsive processing on on-site terminals, it is important to develop technologies for high-performance SLMs that run on edge devices.

In the development and operation of SLMs, simply limiting the model size makes it difficult to achieve sufficient performance. Particularly in highly specialized domains such as manufacturing, there are many technical terms and expressions that are not included in general-purpose models. In highly specialized domains, appropriate domain adaptation is essential for generating appropriate responses tailored to user inquiries and on-site conditions. Therefore, continual pretraining using data from specific domains[(1)] and instruction tuning[(2)], and alignment to achieve responses that are natural and safe for users[(3)]—such fine-tuning methods—need to be appropriately combined.

*1 ChatGPT is a registered trademark of OpenAI OpCo, LLC.

In addition, to operate language models practically in constrained environments such as manufacturing sites, it is also important to achieve optimization while ensuring inference accuracy, considering the balance with response speed and resource usage. In use cases that require an instantaneous response in particular, designs that minimize response latency on site and techniques that ensure stable operation in restricted computing environments are important for achieving both model performance and practicality.

In this paper, we focus particularly on the FA field within the manufacturing industry and, using various documents related to our FA products, an SLM (Fig. 1) was built, and we outline this effort. Chapter 2 outlines representative fine-tuning techniques and practical examples of domain specialization using them. Chapter 3 outlines a practical example of running an SLM on an edge device using OSS. Finally, Chapter 4 summarizes the results of this work and future prospects.

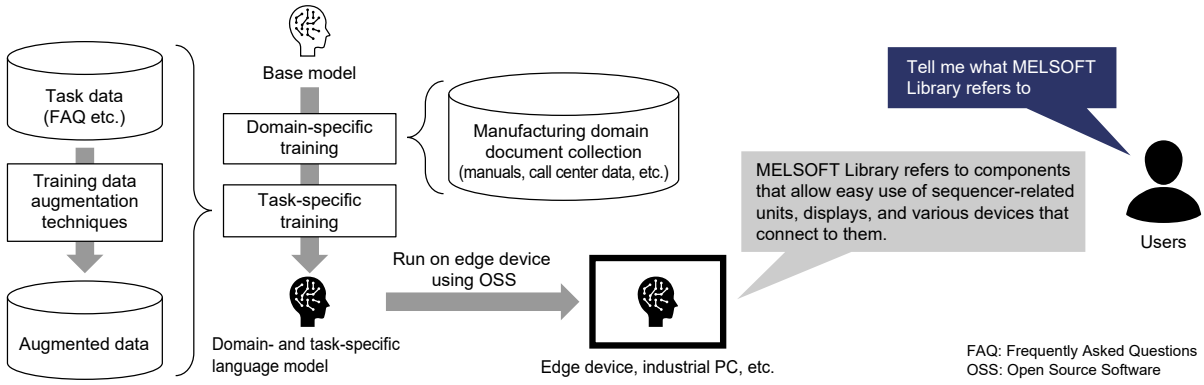

Fig. 1 A domain-specialized SLM for the manufacturing industry running on edge devices

2. Development of an SLM for Domain Specialization for the Manufacturing Industry

LLMs are pretrained using large amounts of general text data, thereby acquiring broad applicability across a wide range of language processing tasks. In practical applications, however, the model needs to be adjusted so that it fits specific languages, domains and use cases. For such post-training, there are several representative methods available, which are used selectively according to their objectives and targets.

In this chapter, we outline representative fine-tuning methods for maximizing language model performance, and then outline the domain specialization for the manufacturing industry approach undertaken in this development.

2.1 Continual pretraining

Continual pretraining is a technique that performs pretraining again on an existing pretrained model using a new corpus. This enables the model to additionally learn vocabulary, expressions and contextual knowledge in specific languages (e.g., Japanese) and domains (e.g., manufacturing industry, healthcare, law).

Because continual pretraining is based on self-supervised learning, it has the advantage that explicit task definitions and annotations are not required, making it easier to leverage unstructured data such as internal documents, equipment manuals and troubleshooting records.

For Japanese and highly specialized domains in particular, there are many linguistic phenomena, vocabulary and variations in notation that general-purpose LLMs do not cover sufficiently, so continual pretraining is an effective approach. On the other hand, to avoid degrading the performance of the existing model, careful balancing with the pretraining data and adjustments to the tokenizer are important.

2.2 Instruction tuning

Instruction tuning is a method for training models to acquire the ability to understand tasks and respond according to instructions. Specifically, supervised learning is performed using pairs of “instructions” and their corresponding “responses”. For example, for an instruction such as “Rewrite the following sentence in polite form,” the model is trained to generate a natural polite sentence without altering the meaning.

This technique is indispensable for enabling users to intuitively leverage the model through chat-style and prompt-style interfaces, and it plays a major role in making interactive use of LLMs possible.

Furthermore, to improve the level of general versatity across diverse tasks, devise ways are needed for collecting and constructing instruction data in various formats.

While there is abundant data for English instruction tuning, high-quality datasets are limited for Japanese, making the use of translated data and the design of Japanese-specific tasks important.

2.3 Alignment

Alignment refers to a set of techniques that align the model's responses with users' values, social norms, and safety considerations, aiming not only to output "correct" responses but to generate responses that are "desirable and appropriate for the user." Responses are also prioritized not only for their quality, but also for safety and ethical considerations. To achieve alignment, there are two main approaches available: learning based on human preferences, and safety tuning.

In learning based on human preferences, pairs of a "more preferred response" and a "non-preferred response" assessed by humans are used, enabling the model to learn a tendency to choose responses that are appropriate for the user. This method is used to evaluate model outputs and reinforce responses with higher quality and consistency. For example, Direct Preference Optimization (DPO)[4] and Proximal Policy Optimization (PPO)[5] are widely used; adjusting model outputs via reward signals makes it possible to improve response quality.

Safety tuning also aims to suppress outputs that include aggressive, inappropriate or dangerous content. In particular, when applied to Japanese, safety needs to be ensured with consideration for cultural backgrounds and social norms that differ from those of English. For example, establishing filtering criteria that account for the linguistic characteristics and social customs of Japanese is required to avoid unintended misunderstandings and inappropriate responses. Such efforts directly improve reliability when integrating models into business systems and public services, and are indispensable in domains where safety is paramount, such as the manufacturing industry.

Thus, the alignment process is an important means of adapting the model's responses to users' expectations and requirements. By achieving appropriate alignment, it is possible to enhance the model's reliability and practicality, enabling the construction of safer and more effective systems.

2.4 Methods for domain specialization for the manufacturing industry

Here we will outline domain specialization for the manufacturing industry using the methods discussed so far. The workflow for domain specialization for the manufacturing industry is shown in Fig. 2. In this initiative, we used various documents related to our factory automation (FA) products and applied the three approaches of continual pretraining, instruction tuning and alignment. This enabled the model to efficiently learn the specialized knowledge and terminology unique to the manufacturing industry, improving performance.

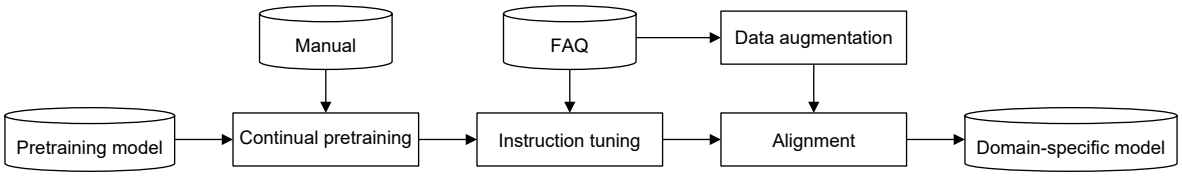

Fig. 2 Flow of domain specialization for the manufacturing industry

As the base pretraining model, a model with 1.8 billion parameters[6] (Model architecture: Llama2[7]) was adopted. In selecting the model size, we considered the balance between computational cost and performance to efficiently conduct training specialized for the manufacturing domain. In manufacturing in particular, building a high-accuracy model while taking resource constraints into account is required, so we selected a model of this size. We also place an importance on increasing model transparency by handling the model's training data in an appropriate manner so that we can explain the model's outputs in the future. From this perspective, we decided to adopt this base model, for which the training data are also publicly available. In the subsequent processing, we applied various domain-specialized methods using this base model.

For continual pretraining, we used manual data related to FA products. This manual data covers a wide range of content, including product specifications, operating procedures and troubleshooting. By retraining

the pretrained model on this specialized corpus, we aimed to efficiently incorporate FA product–specific vocabulary and expressions, as well as structured information.

For instruction tuning, we used FAQ data related to FA products. This data is structured in a format that includes questions about product usage and troubleshooting along with their answers. Instruction tuning enables the model to improve its ability to generate appropriate responses to user questions. In particular, tuning at this stage is essential to provide natural responses to inputs in a question format. Furthermore, to address inquiries specific to manufacturing, we also reviewed the content of the FAQ data and organized it into an appropriate format.

For alignment, as with instruction tuning, we leveraged FAQ data related to FA products. The goal of alignment is to enhance the model's ability to generate preferred responses for users. However, while FAQ data includes questions and their "appropriate answers," we needed to address the absence of "non-preferred responses." Therefore, from the set of answer texts within the existing FAQ data, we extracted answer texts with high text similarity that are similar but not the same, and regarded them as "non-preferred responses" to the same questions as a newly considered data augmentation method. With this method, without performing additional manual annotation or designing "non-preferred responses," we were able to efficiently prepare alignment data pairs from existing domain data.

By combining these methods—continual pretraining, instruction tuning and alignment—we made it feasible to build a model specialized for the manufacturing domain. For the model's performance evaluation, we adopted LLM-as-a-judge using Anthropic Claude-3.7 Sonnet as the evaluation model[(8)] for reference-guided evaluation. For the evaluation data, we used a set of questions that test the correctness of knowledge about our FA products, and for comparison we selected OpenAI GPT-4o, a representative general-purpose LLM on the cloud. For the evaluation data, we compared the responses of the model we developed and those of the comparison model GPT-4o against reference answers and determined correctness. This comparison allows us to evaluate the effectiveness of the developed model with a certain degree of objectivity.

As a result of the evaluation, we confirmed that the model we developed achieved an accuracy rate of 77.24%. GPT-4o, the comparison target, had an accuracy rate of 52.03%. Because this evaluation uses two-choice questions that ask about the correctness of knowledge regarding our FA products, the accuracy rate (chance rate) when answering randomly is expected to be 50%. The accuracy rate of GPT-4o was close to the chance rate. In contrast, the model we developed exceeded the chance rate by 27.24%, demonstrating the effectiveness of the domain-specialized methods examined in this development.

3. Implementing Edge AI by Leveraging OSS

Unlike the conventional approach of running LLMs in the cloud, a domain-specialized SLM is expected to run on edge devices (edge AI) to meet on-site needs for lightweight operation and high responsiveness. By operating the SLM as edge AI, low-latency and privacy-conscious processing becomes possible, and its value is significant across various fields such as smart factories, edge robotics and energy control.

To run a domain-specialized SLM efficiently on an edge device, optimization that balances performance and accuracy within limited hardware resources is essential. As a means of optimization, leveraging a flexible and highly capable OSS is crucial. Our company participates in development alongside world-class engineers in OSS communities that underpin AI development, simultaneously improving our technical capabilities and contributing to society. Specifically, in Apache TVM, an AI compiler OSS[*2] [(9)], our engineers are active in a central role as "committers" with source code edit rights, and in PyTorch[*3], which has become the de facto industry standard as an AI framework, our contributions have also been recognized, and we were selected as finalists for the PyTorch Contributor Awards 2024[(10)].

In this chapter, drawing on our knowledge of these OSS, we describe our proof-of-concept efforts to run an SLM on edge devices. The target devices are the GPU-equipped Jetson[*4] Orin Nano 8GB and the Radxa ROCK 5B 16GB equipped with an NPU (Neural Processing Unit); both were selected with industrial applications in mind. In this study, we examined a method to run an SLM with 3.7 billion parameters (model

*2 Apache TVM is a registered trademark of the Apache Software Foundation.
*3 PyTorch is a registered trademark of the Linux Foundation.
*4 Jetson is a registered trademark of NVIDIA Corp.

architecture: Llama2) on both devices in a memory-efficient and high-speed manner by leveraging OSS. Prior to the evaluation, we confirmed that with common execution methods such as PyTorch's Eager mode, the above SLM could not be run on the target devices due to insufficient memory (required memory: 14.9 GB), underscoring the importance of this effort.

To run an SLM on an edge device, an appropriate OSS for LLM inference first needs to be selected. In selecting OSS, Chapter 2 outlined two requirements: support for the SLM (model architecture: Llama2) outlined there and support for the target devices; we evaluated the usefulness of leveraging each OSS that met these.

On the Jetson Orin Nano 8GB, MLC-LLM (Apache TVM-based), ExecuTorch, ollama, vLLM, IREE, and llama.cpp[11] were selection candidates, and we ultimately adopted llama.cpp. llama.cpp takes a "handcrafting" approach to implementing inference—that is, manually optimizing code for each hardware platform—which offers high flexibility in early development stages and relatively easy adoption; these points led us to choose it. As a result, we were able to run the SLM on the Jetson Orin Nano 8GB and confirmed practical inference speed. The generation speed and memory usage at runtime are shown in Table 1.

Table 1 Results of running the domain-specialized SLM on edge devices (Jetson Orin Nano 8GB)

Target device	Jetson Orin Nano 8GB (GPU-equipped)
SLM to be executed	3.7B-class SLM (architecture: Llama2)
Candidate OSS	MLC-LLM (Apache TVM), ExecuTorch, ollama, vLLM, IREE, llama.cpp
Adopted OSS	llama.cpp
Generation speed	22 tokens/sec
Memory usage	2.4GB

On the Radxa ROCK 5B 16GB, MLC-LLM (Apache TVM-based) and rkllm were candidate selections, but because MLC-LLM did not sufficiently support the target device, we adopted rkllm. The generation speed and memory usage at runtime are shown in Table 2.

Table 2 Results of running the domain-specialized SLM on edge devices (Radxa ROCK 5B 16GB)

Target device	Radxa ROCK 5B 16GB (NPU-equipped)
SLM to be executed	3.7B-class SLM (architecture: Llama2)
Candidate OSS	MLC-LLM (Apache TVM-based), rkllm
Adopted OSS	rkllm
Generation speed	6.8 tokens/sec
Memory usage	4.5GB

In this way, by leveraging optimal OSS for the two target devices, even though general AI execution methods suffer from insufficient memory, we confirmed that an SLM with 3.7 billion parameters, previously impossible to run due to insufficient memory, is executable.

Broadly speaking, there are two approaches to the design of LLM inference frameworks. One is the handcrafting approach, as with llama.cpp adopted here, which achieves a high level of flexibility by directly adjusting the source code. The other is the compiler approach, exemplified by Apache TVM, which automatically generates optimized code tailored to the target device.

The handcrafting approach is flexible in supporting efficient fine-tuning methods such as LoRA (Low-Rank Adaptation), and is relatively easy to adopt. On the other hand, the platforms and devices it can support are limited, and implementation and optimization are required for each target device, leaving challenges in terms of scalability and maintainability.

In contrast, the compiler approach, although currently constrained in its support for LoRA, supports a wide range of platforms, including mobile and browser environments as with MLC-LLM, and is superior from the standpoint of future extensibility and scalability. Furthermore, because code generation can automate optimization for each environment, improvements in maintainability and development productivity can be expected over the long term.

We intend to advance a transition to the compiler approach in order to strengthen our ability to support increasingly diverse device environments and use cases going forward. In addition, regarding the implementation of training optimization features starting with LoRA, we are aiming to expand supported capabilities in collaboration with the OSS community. Leveraging the OSS utilization expertise we have cultivated to date, we will accelerate the practical use and adoption of domain-specialized SLMs in the edge AI field and work to achieve a more sustainable and highly scalable society.

4. Conclusion

These results show that, for developing SLMs specialized for expert and limited domains such as manufacturing, the combination of continual pretraining, instruction tuning and alignment is effective. This approach can also be applied to other manufacturing-related domains and to other industries, and by broadening its scope of application, further performance improvements can be expected.

Future challenges include collecting additional data to further improve model performance and improving alignment methods. In particular, there is a need to refine methods for the automatic generation of non-preferred response and to devise ways to reconcile response diversity and consistency. In addition, for model evaluation methods, approaches need to be considered that use more multifaceted metrics to quantitatively measure practical utility. Moreover, to address a wide variety of use cases, optimization on edge devices needs to be considered. Depending on the use case, higher responsiveness and lower resource usage than currently assumed may be required. Therefore, implementation-level ideas are needed to achieve stable model operation and high responsiveness in constrained computing environments. By continuously working on these challenges, we aim to further advance and put SLMs into practical use in the manufacturing domain.

References

(1) Gupta, K., et al.: Continual Pre-Training of Large Language Models: How to (re) warm your model?, The Thirty-Seventh Annual Conference on Neural Information Processing Systems (2023)
(2) Wei, J., et al.: Finetuned Language Models are Zero-Shot Learners.The Tenth International Conference on Learning Representations (2022)
(3) Bai, Y., et al.: Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback, arXiv, 2204.05862 (2022)
(4) Rafailov, R., et al.: Direct Preference Optimization: Your Language Model is Secretly a Reward Model, Advances in Neural Infor- mation Processing Systems, 36, 53728-53741 (2023)
(5) Schulman, J., et al.: Proximal Policy Optimization Algorithms, arXiv, 1707.0634 7 (2017)
(6) Hugging Face: LLM-jp
https://huggingface.co/llm-jp
(7) Touvron, H., et al.: Llama 2: Open Foundation and Fine-Tuned Chat Models, arXiv, 2307.09288 (2023)
(8) Zheng, L., et al.: Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena, Advances in Neural Information Processing Systems, 36, 46595-46623 (2023)
(9) Chen, T., et al.: TVM: An Automated End-to-End Optimizing Compiler for Deep Learning, 13th USENIX Symposium on Oper- ating Systems Design and Implementation, 578-594 (2018)
(10) PyTorch: Announcing the 2024 PyTorch Contributor Awards
https://pytorch.org/ecosystem/contributor-awards-2024
(11) GitHub: ggml-org/llama.cpp
https://github.com/ggml-org/llama.cpp