Practical viability of translating legacy Fortran code to C++ using large language models

Ren Imai (imai.ren.t6@dc.Tohoku.ac.jp) [1] , Masatoshi Kawai[1] , Keichi Takahashi[2] , Hiroyuki Takizawa[1]
[1]Tohoku University , [2]The University of Osaka

Background and Goal

Accelerating Adoption of C++

- The adoption of C++ in high performance computing (HPC) is expanding, and thus more developers are familiar with C++.
- It is possible to make the most of state-of-the-art features and/or devices.

Difficulties in Fortran code maintenance

- Many scientific and technical computing applications have been written in Fortran.
- Old Fortran code used for many years is becoming legacy code because fewer developers can read and maintain it.

→ The increasing need to migrate from Fortran to C++.

Related Work using LLMs

- Lei et al. translated OpenMP benchmarks written in Fortran 90 to C++.
  - Fine-tuning with Fortran codes can significantly improve the translation performance of LLMs, especially for the LLM not originally trained on Fortran [1].
  - The translation performance is discussed based on CodeBLEU [2].
- Pan et al. analyzed the translation performance of multiple LLMs.
  - Errors in the generated C++ code can be classified into 15 distinct categories [3].

Our Goals

- To quantitatively evaluate the performance of a Code LLM (LLM tuned for code generation) at automatic Fortran-to-C++ translation.
- To discuss whether advanced usage of LLMs can improve the translation accuracy, using three evaluation metrics.

Evaluation Setup

Evaluation Environment

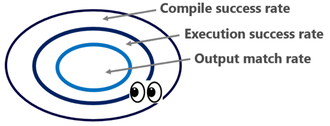
- Model Employed : OpenCoder-8B-Instruct [4] & GPT-5 [5]
- Data : Original Custom-designed to yield a particular output

Sample Dataset	Count
Fortran Exercises (Practice) [5]	45
Custom OpenMP Problems	3
Total	48

Evaluation Dataset	Count
Fortran Exercises (Applied)	35
Polybench [6]	30
Rosetta Code [7]	263
Total	328

Evaluation Method

1. We designed custom prompts based on the translation results from the sample dataset.
2. We evaluated the performance of the evaluation dataset translation across four patterns based on the application of **In-Context Learning** [9] (**ICL**: translation using custom prompts) and **iterative translation** [3](**IT**: re-translating the code when a compile error occurs).
   - OpenCoder was tested three times for each of the four patterns.
   - GPT-5 was evaluated once only for the Base pattern (no ICL or IT).
3. We calculated the compile success rate, execution success rate, and output match rate, which represent the percentage of cases where at least one generated code could be compiled, executed, and produce the correct output, respectively.
   - The output match rate was manually verified by human experts.



Evaluation Results and Discussions

Sample Dataset translation

- The translation could fail due to the differences in language specifications between Fortran and C++
  - The prompt is as follows, and we refer to it as "Base"

Convert the following Fortran code to C++ code step by step

Feature Language	Fortran	C++
Default Array Indexing	1-based	0-based
Code Organization	Modules	Namespaces
Direct Array Operations	Supported	Library-dependent
Parallelism	Built-in (Coarrays)	Library-dependent

⇒ Custom Prompt design

- The Custom prompt is as follows, and we refer to it as "ICL"

Convert the following Fortran code to C++ code step by step.
Make sure to follow these rules:
1. In Fortran, array indices start at 1, while in C++, they start at 0
2. In Fortran, calculations can be applied directly to arrays or sub-arrays, while in C++, they must be performed element by element in loops.
3. Use namespaces in C++ code only if modules are used in Fortran code
4. Use #pragma omp critical for output in C++ code only if openmp is used in Fortran code

- With the custom prompt, codes in the sample dataset are translated again, and all the generated C++ codes can finally produce correct output

Evaluation Dataset translation

- The "ICL + IT" method outperformed the "Base" method on all metrics.
- GPT-5 achieved the best output match rate on the Fortran Exercises (Applied) and Rosetta Code datasets, but produced no correct translations for Polybench.
  - Since all the datasets are available online, GPT-5 might have already been trained with Fortran Exercises and Rosetta, but not with Polybench by chance

LLM	translation Method	Fortran Exercises (Applied)			Polybench			Rosetta Code		
		compile	execution	output	compile	execution	output	compile	execution	output
OpenCoder	Base	88.6	88.6	77.1	50.0	43.3	3.33	77.9	74.1	46.4
	ICL	88.6	88.6	82.9	60.0	50.0	23.3	74.1	70.3	47.1
	Base + IT	91.4	91.4	80.0	76.7	70.0	10.0	85.9	81.0	49.8
	ICL + IT	91.4	91.4	82.9	80.0	70.0	30.0	84.4	82.1	51.3
GPT-5	Base	88.6	88.6	88.6	3.33	0.00	0.00	86.6	84.8	78.6

Conclusions

- We have evaluated the performance of Code LLMs at automatic Fortran-to-C++ translation, and discussed how to quantify and improve the translation performance
  - The evaluation results demonstrate that use of ICL and IT is effective to improve the performance in terms of all three metrics defined.
  - GPT-5 works perfectly for some datasets, but fails for Polybench(output match rate = 0%)
- Since we found ICL and IT useful for OpenCoder, appropriate design of those techniques may also be effective for GPT-5.

References

- [1] Bin Lei, Caiwen Ding, et al. Creating a dataset for high-performance computing code translation using LLMs: A bridge between openmp fortran and C++. arXiv preprint arXiv:2307.07686, 2023.
- [2] Ren et al. CodeBLEU: a method for automatic evaluation of code synthesis. arXiv preprint arXiv:2009.10297, 2020.
- [3] Rangeet Pan et al. Lost in translation: A study of bugs introduced by large language models while translating code. IEEE/ACM 46th International Conference on Software Engineering (ICSE 2024), pages 1–13, 2024.
- [4] Huang et al. OpenCoder: The open cookbook for top-tier code large language models, 2024. URL https://arxiv. org/abs/2411.04905, 2411:04905
- [5] OpenAI. Gpt-5 system card. https://cdn.openai.com/pdf/8124a3ce-ab78-4f06-96eb-49ea29ffb52f/gpt5-system-card-aug7.pdf, 2025
- [6] Takanobu Amano. Fortran Exercises (Geophysics Practicum), https://amanotk.github.io/fortran-resume-public/
- [7] https://www.cs.colostate.edu/~pouchet/software/polybench/
- [8] Mike Gunderloy. Rosetta Code. https://rosettacode.org/wiki/Category:Fortran.
- [9] Clyde Highmore. In-context learning in large language models: A comprehensivesurvey. 2024.