Translating scientific workflows from nextflow to snakemake using Large Language Models

Bachelor Thesis Expose

Mario Jäckle
Supervisor: Prof. Ulf Leser

September 2023

Introduction

Scientific workflows are a way to represent and execute complex scientific tasks or data analysis pipelines in a systematic and reproducible manner. They provide a structured and organized approach to managing data, conducting analyses, and automating repetitive tasks, making them widely used in bioinformatics, computational biology, and other scientific fields. (Ahmed et al, 2021).

These workflows are defined inside a so-called workflow management system (WfMS). These WfMS are software tools that automate and oversee the design, execution, and coordination of complex sequences of tasks or processes within an organization or project. (Barker, A. et al, 2008). There are a range of different WfMS like nextflow[1], snakemake[2] and Apache Airflow[3].
Each of these is equipped with its unique Domain-Specific Language (DSL) to define their workflow pipelines (Ahmed et al, 2021). As these DSLs come in a wide variety of different workflow languages the defined workflows of one WfMS are not interchangeable and usable with other WfMS.

Restricted runtime environments or the desire to use a module or (sub-)workflow within a none-matching WfMS can make it necessary to convert workflows defined in one WfMS to another. As underlying project structures and DSL languages are not easily convertible, this is not a trivial task and consumes a lot of time and effort.

The most recent generation of Large Language Models (LLMs) has exhibited remarkable capabilities in generating, debugging and translating code (Chen, M et al, 2021). This development prompts the exploration of whether LLMs can be harnessed to facilitate the seamless translation of workflows from one WfMS to another.

[1] Nextflow (2023), https://www.nextflow.io, access date: 18.08.2023
[2] GitHub: Snakemake (2023), https://github.com/snakemake/snakemake, access date: 18.08.2023
[3] Apache Airflow (2023), https://airflow.apache.org/, access date: 18.08.2023

Background

nf*core[4]

nf*core is a collaborative effort within the field of bioinformatics, focusing on the development and standardization of reusable, reproducible, and user-friendly Nextflow-based pipelines. These pipelines encompass diverse data analysis processes spanning genomics, transcriptomics, proteomics, and more (Ewels et al. 2020). nf*core aims to foster the creation of high-quality and consistently structured workflows.
At the time of writing the nf*core repository contains 86 pipelines, 992 modules (self-contained tasks) and 44 subworkflows (common chains of tasks) [5] with pipelines containing 17 modules and 6 subworkflows on average. Furthermore nf*core provides a repository with test data for all pipelines and modules. [6]

Large language models

A Large Language Model (LLM) is a transformer based neural network (Vaswani, A. et al, 2017) trained to understand and generate human-like text through the application of deep learning techniques. These models are trained on vast datasets containing diverse examples of human language use, allowing them to learn intricate patterns of syntax and semantics (Radford et al, 2019). This enables LLMs to perform a variety of language-related tasks with remarkable proficiency, including natural language understanding, text generation, translation, summarization, and more. (Radford et al, 2019) Prominent examples of LLMs include OpenAI's GPT-4[7], Facebook's Llama (Touvron, H et al, 2023) or Google's BERT (Devlin, J et al, 2019). Such models leverage a transformer architecture, which has revolutionized the field of natural language processing (NLP) by significantly improving the quality of language-related tasks. (Vaswani, A. et al, 2017)

Prompts

In a LLM context, a prompt "is a set of instructions provided to an LLM that programs the LLM by customizing it and/or en- hancing or refining its capabilities." (White, J. et al. 2023). This prompt is mostly provided in the form of natural language containing instructions, examples and/or input data.

[4] Nf-core (2023), https://nf-co.re/, access date: 17.08.2023
[5] Nf-core (2023), https://nf-co.re/, access date: 19.08.2023
[6] GitHub nf*core 2023, https://github.com/nf-core/test-datasets/tree/sarek, access date: 19.08.2023
[7] OpenAI (2023), https://openai.com/gpt-4, access date 17.08.2023

Goal

Nextflow and Snakemake represent two commonly utilized WfMS having a substantial number of public GitHub repositories (599[8] and 484[9]). Nextflow employs Groovy as its DSL (DI Tommaso et al, 2017), while Snakemake relies on Python (Köster et al, 2012). Consequently, the dissimilarity in their DSLs renders them non-trivially interchangeable, complicating the process of transitioning between the two platforms.

While there are various examples of transcompiler (source-to-source compilers), they either rely on handcrafted rules to translate from one language to another, or are based on supervised-trained AI models (Roziere et al, 2020). None of which currently supports the translation of Nextflow to snakemake.

The primary goal of this thesis is to assess the capabilities of a specific LLM, namely GPT-4, in translating Nextflow workflows to Snakemake, thereby investigating the feasibility of leveraging LLMs for such conversions. It will be analyzed in what way LLMs can and should be harnessed to do this kind of code conversions, what prompts achieve the best results and what other steps need to be taken to reach a satisfying translation.

Ideally this thesis also results in a script which can be used to (semi-)automate such conversions and reduce the manual effort when converting from one WfMS to another.

Method

As a first step of the thesis aims to demonstrate the feasibility of translating Nextflow workflows to Snakemake using Large Language Models (LLMs). To achieve this, a simple Nextflow workflow accompanied by test data for input and output will be selected. Through the utilization of LLMs, the translation process will be performed and subsequently validated against the test data.

The next step is to translate more complex Nextflow workflows. For this a set of encapsulated nf*core modules will be selected. Each module in this set will be subjected to translation using LLMs, followed by validation against its test data.

Simultaneously various prompts will be developed, compared and refined to identify which instructions can be used to omit the most occurring translation errors and provide the best overall results. The resulting prompts will be integrated in a script to automate the translation and validation procedures and will later be used to simplify the processing of complete nf*core workflows.

[8] GitHub (2023), https://github.com/topics/nextflow, access date: 27.07.2023
[9] GitHub (2023), https://github.com/topics/snakemake, access date: 27.07.2023

This prompts and script will then be utilized to transform several complete nf*core pipelines, each presenting varying levels of complexity. If necessary the previously developed prompts and script will be further refined and adjusted for any special requirements of the nf*core workflow structure.

The results will either be analyzed in a qualitative and/or quantitative manner, depending on what is more applicable to the result.
The qualitative analyses will contain checking if valid, runable code is generated and if the code produced is equivalent to the input code. The latter can be verified by feeding the same test data to the original code as well as to the generated code and comparing the results of both.
A quantitative analysis would consist of pinpointing in which sectors (e.g. syntax, logic, file references) the transitions commonly fail, grouping them in this error classes and counting their occurrences.

Literature

Ahmed, A. E., Allen, J. M., Bhat, T., Burra, P., Fliege, C. E., Hart, S. N., Heldenbrand, J. R., Hudson, M. E., Istanto, D. D., Kalmbach, M. T., Kapraun, G. D., Kendig, K. I., Kendzior, M. C., Klee, E. W., Mattson, N., Ross, C. A., Sharif, S. M., Venkatakrishnan, R., Fadlelmola, F. M., & Mainzer, L. S. (2021). Design considerations for workflow management systems use in production genomics research and the clinic. Scientific Reports, 11(1), 1–18. URL: https://doi.org/10.1038/s41598-021-99288-8

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever (2019). Language Models are Unsupervised Multitask Learners.
Whitepaper @ https://openai.com/research/better-language-models (access date 23.08.2023)

Barker, A., van Hemert, J. (2008). Scientific Workflow: A Survey and Research Directions. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Wasniewski, J. (eds) Parallel Processing and Applied Mathematics. PPAM 2007. Lecture Notes in Computer Science, vol 4967. Springer, Berlin, Heidelberg. URL: https://doi.org/10.1007/978-3-540-68111-3_78

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. de O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., … Zaremba, W. (2021). Evaluating Large Language Models Trained on Code. URL: http://arxiv.org/abs/2107.03374

DI Tommaso, P., Chatzou, M., Floden, E. W., Barja, P. P., Palumbo, E., & Notredame, C. (2017). Nextflow enables reproducible computational workflows. Nature Biotechnology, 35(4), 316–319. URL: https://doi.org/10.1038/nbt.3820

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference, 1(Mlm), 4171–4186.

Ewels, P. A., Peltzer, A., Fillinger, S., Patel, H., Alneberg, J., Wilm, A., Garcia, M. U., Di Tommaso, P., & Nahnsen, S. (2020). The nf-core framework for community-curated bioinformatics pipelines. Nature Biotechnology, 38(3), 276–278. URL: https://doi.org/10.1038/s41587-020-0439-x

Köster, J., & Rahmann, S. (2012). Snakemake-a scalable bioinformatics workflow engine. Bioinformatics, 28(19), 2520–2522. URL: https://doi.org/10.1093/bioinformatics/bts480

Roziere, B., Lachaux, M. A., Chanussot, L., & Lample, G. (2020). Unsupervised translation of programming languages. Advances in Neural Information Processing Systems, 2020-Decem.

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. (2023). LLaMA: Open and Efficient Foundation Language Models. URL: http://arxiv.org/abs/2302.13971

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 2017-December(Nips), 5999–6009.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017) 'Attention is all you need', Advances in Neural Information Processing Systems, 2017-Decem(Nips), pp. 5999–6009.

White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., & Schmidt, D. C. (2023). A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT. URL: http://arxiv.org/abs/2302.11382