Humboldt-Universität zu Berlin
Institut für Informatik

Creating a question answering dataset for scientific workflow code

Study Project Expose

Simon Bosse

January 27, 2025

1 Introduction

The analysis of vast amounts of scientific data often requires complex processing chains using different domain-specific tools. To process large amounts of data in reasonable time spans, scientific data analysis is often executed on distributed systems. This also introduces challenges such as scalability and reproducibility. To deal with these challenges, workflow systems can be used, allowing users to abstractly define their data analysis step by step in domain-specific languages. Scientific Workflow Management Systems (SWMS) such as Nextflow [6] or Snakemake [10] then take care of workflow execution.

The development of scientific workflows can be challenging, since it requires both substantial domain knowledge and considerable software engineering skills. An alternative to developing workflows from scratch is using best practice workflows such as those curated by the `nf-core` [7] community. But such workflows are often highly standardized and integrate many analysis options. This also introduces significant code complexity. Execution, adaptation and extension of these workflows can hence be challenging for developers. A recent study identified code-related errors as a common cause of workflow execution failure [15].

To aid software developers, LLM-based tools such as GitHub copilot[1] can be used. However, these tools are mostly tailored to code generation and completion in traditional programming languages. In contrast, systems designed for code-based question answering (QA) can correctly answer questions about given programming code, for example about the codes behaviour, or implementation details. QA systems for code are rather rare, which can also be attributed to a lack of available QA datasets on programming code [4]. This is despite the fact that performant code QA systems could significantly improve developers code understanding, and thereby facilitate tasks such as code maintenance or extension.

These aspects are also relevant to workflow languages such as Nextflow. Especially non-coding experts could benefit from systems reliably answering their questions regarding Nextflow code: They could potentially help developers with understanding errors or unexpected behaviour.

[1] https://github.com/features/copilot

2 Goals

This work aims to create a dataset for question answering on Nextflow programming code by providing question-answer pairs on reference Nextflow workflows collected from `nf-core` and GitHub[2]. Such a dataset could be used to analyze how well modern neural models understand and explain workflow code. We seek to extensively annotate each question-answer pair with:

1. the Nextflow concepts that are relevant to the question, for example channels, processes or workflows,
2. the question type, such as comparison questions or boolean questions,
3. and the lines of code in the corresponding workflow serving as supporting facts, i.e. they are relevant to correctly answer the question.

A more detailed description of the different annotations as well as an exemplary question-answer pair can be found in Section 4. The annotations should allow for a more precise evaluation of model's performance on the dataset. We also aim to benchmark the question answering task of the dataset using language models such as CodeBERT [8]. We plan to both generate free-form answers and identify relevant lines of code (supporting facts).

3 Related Work

Question answering is the task of answering a question using given context information. In the field of natural language processing, models are often exposed to QA problems to evaluate their ability to correctly process textual information in different contexts, such as Wikipedia excerpts or reference web pages retrieved from search engines. Over the past years, researchers created various QA datasets with diverse characteristics.

One of the most prominent QA datasets is **SQuAD** [18]. Its questions can be answered by identifying one continuous span of text in the corresponding context, hence not forcing models to actually generate an answer. **HotpotQA** [20] is a multi-hop QA dataset. To answer a question, multi-hop reasoning is needed, i.e. connecting information across multiple documents.

[2]https://github.com/

A question answering dataset constructed using Bing search queries is **MS MARCO** [3]. For each question, it provides an answer (e.g. numerical or free-form text answers) along with annotations on the passages of multiple reference documents relevant to generate the answer. Similarly, the **Natural Questions** [11] dataset consists of real-world questions from Google. As answers, it provides long answers in the form of relevant paragraphs in a Wikipedia article, and if applicable, a short answer consisting of one or multiple short text spans.

The **QuALITY** [16] dataset is a QA dataset on long contexts. Since most performant models are limited to process a rather small amount of tokens at once, they find models to perform poorly on QA with long contexts.

Other examples of QA datasets are **ToolQA** [21], which evaluates a systems ability to make use of external tools, and **TheoremQA** [5], testing how succesful models apply scientifical theorems to given problems. The **VQA** [2] dataset is a benchmark for visual question answering.

There also exist several code-specific QA datasets. **CodeQA** [14] contains questions on snippets of python and Java programming code, which requires systems to process both natural language and programming code. Answers are mostly given by rather short sentences or phrases. As context, usually only the relevant code is provided. Similar to other state-of-the-art QA datasets, it includes a wide range of question types, requiring systems to process comparison, yes/no, multiple choice and open questions.

Similarly to SQuAD, answers in **CodeQueries** [19] are given by text spans in the context. The dataset consists of semantical queries with code snippets as context, and similar to HotpotQA, the answer span and its supporting facts need to be identified in the context. **CS1QA** [12] consists of questions on python snippets, collected from an introductory programming class. **CoReQA** [4] and **CodeRepoQA** [9] contain questions on GitHub repositories, and have been constructed from issues and comments within the respective repositories.

4 Dataset design

In QA, there exist several approaches to generate new datasets. Often, question-answer pairs are generated by humans given one or more context documents [5, 18, 20]. Sometimes, only answers are annotated by humans, while questions are collected from real-world users, e.g. via search engines [3, 11].

Question

In what cases is the process `MEGAHIT` skipped?

Nextflow code

```
[...]
if (!params.skip_megahit) {
    MEGAHIT(ch_short_reads_grouped)
    ch_megahit_assemblies = MEGAHIT.out.contigs.map { meta, assembly ->
        def meta_new = meta + [assembler: 'MEGAHIT']
        [meta_new, assembly]
    }
    ch_assembled_contigs = ch_assembled_contigs.mix(ch_megahit_assemblies)
    ch_versions = ch_versions.mix(MEGAHIT.out.versions.first())
}
[...]
```

Answer

if `ch_short_reads_grouped` is empty or if the parameter `skip_megahit` is set to true

Figure 1: Self-constructed exemplary question-answer pair

Questions can also be generated template-based [21], rule-based [14] or using LLMs [4], which significantly reduces the human workload.

For this thesis, we plan to create question-answer pairs manually for given contexts (i.e., Nextflow workflows). This will distinguish the dataset from other code QA datasets, that relied on template- and rule-based approaches [14] or used LLMs for generation [4].

Real-world questions from platforms like GitHub or Stack Overflow[3] may be used as starting points, but automatically collecting user questions from these platforms is not trivial: Many public questions on Nextflow relevant to our setting deal with execution problems instead of coding. Content-wise, the question should orientate towards queries developers may ask about their workflow code.

[3] https://stackoverflow.com/

An example of a question-answer pair with its corresponding context is shown in Figure 1. The shown code snippet is only a part of the `nf-core` workflow mag which serves as context to the given question. To generate the correct answer, a model would need to identify where in the code the process `MEGAHIT` is called, and then understand the control flow surrounding it as well as applying Nextflow specifications: Processes are also skipped if one input is empty. This is a question a researcher may ask if he does not understand why the task is skipped when executing the workflow.

To create a diverse QA dataset on Nextflow code which is also coherent and broadly annotated, there are further design decisions to be taken: about the set of context documents, the types of questions and answers, and what information should be annotated. These aspects are described in the following sections.

As context documents that question-answer pairs are associated to, we choose to use complete workflows from `nf-core` and other popular Nextflow workflows collected from GitHub. Since `nf-core` workflows are highly standardized, we want to prevent dependencies by incorporating other workflows too. Also, most `nf-core` workflows are highly modularized. Since we want to use complete workflows, we plan to parse each modularized workflows into one file.

Similar to QuALITY [16], we expect using long contexts to increase the difficulty of the QA task since models also need to identify relevant passages. Also, some properties of workflows are only captioned in long-range dependencies (e.g. control flow).

4.1 Question annotation

We aim to label the question-answer pairs in the dataset with information on the Nextflow concepts that are relevant to answer it. This way, we seek to enable a precise evaluation of models performance on the dataset: By reporting performance not only on the whole dataset but also on subsets concerning specific Nextflow concepts, it will be easier to analyze in which areas a model has the most significant need for improvement. Nextflow concepts to be annotated are described in the following.

- **Processes** are the modular units encapsulating the single scripts/-command line tools running within a Nextflow workflow. Therefore, processes include input/output declarations as well as the script which

the process encloses. Additionally, variables can be defined in a process using groovy code.

- **Channels** are used for the communication between processes. Process inputs and outputs are provided respectively emitted via channels, and channels can also be created from values.
- **Workflows** are functions defining the dataflow of the channels and processes contained in the Nextflow code. Workflows can be modularized, i.e. it is also possible to create subworkflows.

There are several other Nextflow concepts which may be worth labeling, including

- **conditional execution** within workflows or processes,
- **channel operators** used to modify channels,
- **inputs/outputs** of processes and workflows
- and the **groovy code** included in processes.

Lastly, a question may require **tool-specific knowledge** about the tools encapsulated in the Nextflow processes. Examples for this could be questions about what output a specific tool produces (a task humans could solve by looking up the tools documentation), or what task a specific process (i.e. an encapsulated bio-tool) performs.

4.2 Question and answer types

Next to annotating information on the domain-specific concepts needed to answer a question, we also seek to label the dataset with information on the types of question-answer pairs.

There are several ways to classify the data points in QA datasets. A popular strategy is to categorize by question word [1, 14, 20]. QA pairs can also be grouped by their answer, e.g. by different types of entities, answer clauses and numerical answers [3, 18].

Both these approaches can give valuable insights into the general characteristics of a QA dataset. We therefore plan to provide statistics on both for the complete dataset. However, for labeling the questions-answer pair type,

we choose broader categories of questions and annotate whether a question is open-ended, numerical, a comparison, multiple choice or a yes/no question. We plan to keep answers rather short, similar to the example in Figure 1.

However, it may later be of interest to examine if a system successfully identifies the lines of code relevant to answer a specific question. To enable an evaluation of this sub task, we choose to annotate the relevant lines of code for each question (similar to supporting facts [19, 20]).

The example question from Figure 1 would be annotated with the labels **workflow** and **conditional execution**. It also is an open-ended question. The supporting fact lines are marked in Figure 2.

Nextflow code

```
[...]
if (!params.skip_megahit) {
    MEGAHIT(ch_short_reads_grouped)
    ch_megahit_assemblies = MEGAHIT.out.contigs.map { meta, assembly ->
        def meta_new = meta + [assembler: 'MEGAHIT']
        [meta_new, assembly]
    }
    ch_assembled_contigs = ch_assembled_contigs.mix(ch_megahit_assemblies)
    ch_versions = ch_versions.mix(MEGAHIT.out.versions.first())
}
[...]
```

Figure 2: Question context from Figure 1 with supporting facts in red

5 Evaluation

Summarizing statistics describing the dataset, e.g. about the distributions of the aforementioned annotations, question types and such will be provided. For validating the annotated answers, we plan to collect annotations of multiple experts for a subset of the dataset. These annotations will then be used to calculate inter annotator agreement measures. Furthermore, the QA dataset will be benchmarked using modern large language models. We aim to use these language models for both the answer generation and the identification of supporting facts.

For other datasets with short free-form answers [3, 14], researchers evaluated a models performance using metrics like BLEU [17] or ROUGE-L [13]. When assessing a systems ability to identify the lines of codes that serve as supporting facts, metrics like F1 score and EM are used. Aggregating results by labels will then allow for a detailed analysis of a models weaknesses when answering questions on Nextflow code.

References

[1] A. M. N. Allam and M. H. Haggag. The question answering systems: A survey. *International Journal of Research and Reviews in Information Sciences (IJRRIS)*, 2(3), 2012.

[2] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh. VQA: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433, 2015.

[3] P. Bajaj, D. Campos, N. Craswell, L. Deng, J. Gao, X. Liu, R. Majumder, A. McNamara, B. Mitra, T. Nguyen, et al. MS MARCO: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016.

[4] J. Chen, K. Zhao, J. Liu, C. Peng, J. Liu, H. Zhu, P. Gao, P. Yang, and S. Deng. CoReQA: Uncovering potentials of language models in code repository question answering. *arXiv preprint arXiv:2501.03447*, 2025.

[5] W. Chen, M. Yin, M. Ku, P. Lu, Y. Wan, X. Ma, J. Xu, X. Wang, and T. Xia. TheoremQA: A theorem-driven question answering dataset. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7889–7901, 2023.

[6] P. Di Tommaso, M. Chatzou, E. W. Floden, P. P. Barja, E. Palumbo, and C. Notredame. Nextflow enables reproducible computational workflows. *Nature biotechnology*, 35(4):316–319, 2017.

[7] P. A. Ewels, A. Peltzer, S. Fillinger, H. Patel, J. Alneberg, A. Wilm, M. U. Garcia, P. Di Tommaso, and S. Nahnsen. The nf-core framework for community-curated bioinformatics pipelines. *Nature biotechnology*, 38(3):276–278, 2020.

[8] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang, et al. CodeBERT: A pre-trained model for programming and natural languages. *arXiv preprint arXiv:2002.08155*, 2020.

[9] R. Hu, C. Peng, J. Ren, B. Jiang, X. Meng, Q. Wu, P. Gao, X. Wang, and C. Gao. CodeRepoQA: A large-scale benchmark for software engineering question answering. *arXiv preprint arXiv:2412.14764*, 2024.

[10] J. Köster and S. Rahmann. Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics*, 28(19):2520–2522, 2012.

[11] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

[12] C. Lee, Y. Seonwoo, and A. Oh. CS1QA: A dataset for assisting code-based question answering in an introductory programming course. *arXiv preprint arXiv:2210.14494*, 2022.

[13] C.-Y. Lin. ROUGE: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[14] C. Liu and X. Wan. CodeQA: A question answering dataset for source code comprehension. *arXiv preprint arXiv:2109.08365*, 2021.

[15] N. Maligeay, N. Bossut, and K. Belhajjame. Why do scientific workflows still break? In *Proceedings of the 36th International Conference on Scientific and Statistical Database Management*, pages 1–4, 2024.

[16] R. Y. Pang, A. Parrish, N. Joshi, N. Nangia, J. Phang, A. Chen, V. Padmakumar, J. Ma, J. Thompson, H. He, et al. QuALITY: Question answering with long input texts, yes! *arXiv preprint arXiv:2112.08608*, 2021.

[17] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[18] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. SQuAD: 100,000+ questions for machine comprehension of text. *arXiv e-prints*, pages arXiv–1606, 2016.

[19] S. P. Sahu, M. Mandal, S. Bharadwaj, A. Kanade, P. Maniatis, and S. Shevade. CodeQueries: A dataset of semantic queries over code. In *Proceedings of the 17th Innovations in Software Engineering Conference*, pages 1–11, 2024.

[20] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov, and C. D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018.

[21] Y. Zhuang, Y. Yu, K. Wang, H. Sun, and C. Zhang. ToolQA: A dataset for llm question answering with external tools. *Advances in Neural Information Processing Systems*, 36:50117–50143, 2023.