

Towards Transparency and Knowledge Exchange in AI-assisted Data Analysis Code Generation

Robert Haase^{1,2,3,✉}

¹Data Science Center, Leipzig University, Humboldtstraße 25, 04105 Leipzig, Germany

²Center for Scalable Data Analytics and Artificial Intelligence (ScaDS.AI) Dresden / Leipzig

³National Research Data Infrastructure for Microscopy and BioImage Analysis, NFDI4BioImage

The integration of Large Language Models (LLMs) in scientific research presents both opportunities and challenges for scientists. Key challenges include ensuring transparency in artificial intelligence (AI)-generated content and facilitating efficient knowledge exchange among researchers. These issues arise from the intransparent nature of AI-driven code generation and the informal sharing of AI insights, which may hinder reproducibility and collaboration. This paper introduces git-bob, an innovative AI-assistant designed to address these challenges by fostering an interactive and transparent collaboration platform within GitHub and GitLab. While enabling seamless dialogue between multiple humans and multiple LLMs in one thread, git-bob ensures that AI-generated contributions are transparent and reproducible. Moreover, it supports collaborative knowledge exchange, enhancing the interdisciplinary dialogue necessary for cutting-edge research. The open-source nature of git-bob further promotes accessibility and customization, positioning it as a vital tool in employing LLMs responsibly and effectively within scientific communities.

Data Analysis, LLM, Code Generation

Correspondence: robert.haase@uni-leipzig.de, ORCID: 0000-0001-5949-2327

Introduction

Generative artificial intelligence (AI) and Large Language Models (LLMs) in particular are changing the way we do data science. Most prominently, scientists use the technology for interacting with scientific data (1), answer data analysis questions (2, 3), generate data analysis code (4–6), and [re-]write scientific manuscripts (7). Unfortunately, the prompts sent to LLMs are commonly not conserved, and thus, at the time of publication, it might be hard to differentiate human-made and AI-generated parts of the scientific work. A professional peer-review system, for documenting how LLM-generated code was prompted for, and which human reviewed it, is not established in contemporary scientific culture. However, such systems do exist for collaborative code editing involving multiple humans. For example, the online platform github.com is well-established in the open-source software community for discussing issues and potential solutions, building code together, and for peer-reviewing contents. As it was shown before that LLMs can solve real-world GitHub issues (8), developing an AI-assistant that interacts with humans directly within the Github platform is the obvious next step. I am presenting git-bob, an implementation of an LLM-based AI-assistant that can respond to

GitHub issues, discuss potential solutions with humans iteratively, write code for them, and submit it as pull-request to be reviewed by humans. It is technically similar to various online services for data analysis such as the OpenAI ChatGPT Data Analyst or Github Copilot Workflows, with three major differences: 1) Multiple humans can interact with git-bob in one communication thread. This allows bringing together domain specialists, such as life scientists, data-analysts and the AI-assistant in one discussion, stimulating knowledge exchange on how to interact properly with the AI-assistant. 2) Discussions with git-bob and resulting code modifications are conserved in an online-platform that others can read and follow, making the interaction with the AI-assistant fully transparent, and 3) git-bob is open-source. Other developers can read its built-in system prompts and modify them to their needs. Git-bob's source code is available online: <https://github.com/haesleinhuepf/git-bob>.

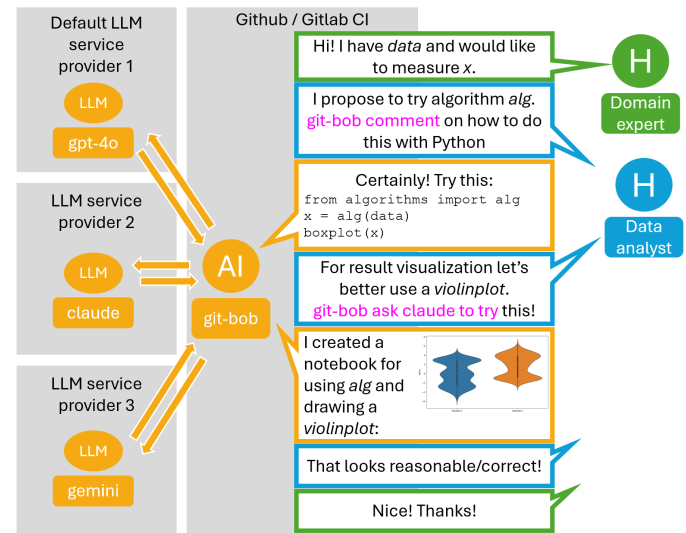


Fig. 1. Schematic view of the interaction with git-bob: In one discussion thread, multiple humans (H) can interact with the AI-assistant which is installed in Github or Gitlab Continuous Integration (CI) infrastructure. Depending on how it is triggered, the AI-assistant may use LLMs from multiple LLM service providers in the background.

Features and limitations

A schematic workflow involving git-bob is drawn in Figure 1: a domain expert opens an issue, a kind of discussion thread, on a repository on github.com, where git-bob is installed. A repository member can then add more information to the request and trigger git-bob to answer by writing a command such as “git-bob comment”. If externals try so, an automatic response will inform them that only repository members are allowed to trigger git-bob because running git-bob may cause costs for repository owners. Once triggered, git-bob will use an LLM to respond to the question, potentially including a code snippet and resulting plots or images. Users and the AI-assistant can then discuss back and forth until some potential solution is reached. This way, good-scientific-practice can be maintained by involving not just domain experts but also data analysis experts in the discussion. Optionally, git-bob can then be asked to implement the solution and send a GitHub pull-request, another kind of discussion thread, but accompanied by file modifications to the repository, e.g. including a Jupyter Notebook containing the previously discussed code solution to a given issue. A human would need to review this pull-request and merge it into the code base of the repository. Git-bob also has the capability to review pull-requests originating from humans, but it is not allowed to merge them. This reflects established practices in science, where eventually a scientist is responsible for data analysis code that becomes part of the project. Common tasks git-bob is capable of are: 1) Giving advice on how to solve a data analysis or data visualization task (Supplementary Figure S1) 2) The assistant can support users of open source libraries by providing advice and code examples, as shown in Supplementary Figure S2. In case the assistant is not sure about the answer, it is capable of forwarding the question to a human (Supplementary Figure S3). 3) It can be used to document code (Supplementary Figure S4). Such a task can be time-consuming when performed without AI-assistance, which can generate documentation for multiple Python functions in seconds to minutes. 4) it can analyse data in the repository directly, e.g. summarize and plot data in CSV files (Supplementary Figure S5). 5) If manuscript files are stored in a github repository, e.g. in latex format, git-bob can assist in writing. For example, the abstract for this manuscript was written by the AI-assistant and this is documented transparently as shown in Supplementary Figure S6.

A highlight of git-bob is that a local installation is not required. Git-bob is implemented as GitHub workflow or Gitlab pipeline, which can be installed by uploading a configuration file to a repository and setting access rights. It is compatible and was tested with the commercial LLMs OpenAI’s GPT4-omni, Anthropic’s Claude 3.5 Sonnet, Google Gemini 1.5 Pro 002, and freely available models hosted on Github Models Marketplace. Git-bob reports which model was used in all of its messages, as good scientific practice suggests. Obviously, the communication with the selected LLM is transmitted to the service provider, including source code files from the repository and images provided with the github issue. Hence, users are recommended to not submit

any personal or sensitive information. When writing data analysis code, git-bob is intrinsically limited by the capabilities of the used LLM. For example, it has been shown that state-of-the-art (SOTA) LLMs can solve bio-image analysis questions by generating functionally correct code just above 50% of tested cases (5). This fundamental limitation may disappear when improved LLMs are published. For now, it can be evaded by the humans guiding the AI-assistant in multi-turn interactions towards a workable solution. Further technical limitations arise from prompt-length limitations of the underlying LLMs. When modifying or generating a file, these files must be below specified limits, e.g. GPT4-omni has 128k tokens input and 16k output tokens as limit (1 token \approx approx. 3/4 words). Also when processing data, limitations of the GitHub IT infrastructure have to be considered: Git-bob executed in public repositories runs on virtual machines with 4 CPU cores, 16 GB of RAM and 14 GB of SSD storage. In private repositories, only 2 CPU cores and 7 GB RAM are available (9). More capable systems are available on a paid basis. By installing git-bob in an institutional Gitlab server, users can setup freely chosen hardware to run git-bob on.

Conclusion

LLMs are being integrated in scientific workflows unavoidably, but commonly it is not documented how they were employed, also because of lack of tools conserving this information conveniently. If the scientific community documented how they prompted LLMs like they document how data analysis software was used, we could learn from each other how to prompt efficiently and responsibly. To overcome current limitations, I propose git-bob, an LLM-based AI-assistant embedded in the Github platform. It enables scientists to interact with an LLM via Github Issues and Pull-Requests offering new ways for implementing good scientific practice for the conservation of discussions between humans and AI when they are working on projects together.

ACKNOWLEDGEMENTS

I would like to thank Elena Katharina Nicolay (UFZ Leipzig) for testing git-bob in its early days and for providing constructive feedback on the manuscript. I also would like to thank Volker Hilsenstein for pushing for Gitlab interoperability. I am also glad that I could use the bioRxiv manuscript template by Ricardo Henriques, which is licensed CC-BY 4.0. I acknowledge the financial support by the Federal Ministry of Education and Research of Germany and by Sächsische Staatsministerium für Wissenschaft, Kultur und Tourismus in the programme Center of Excellence for AI-research “Center for Scalable Data Analytics and Artificial Intelligence Dresden/Leipzig”, project identification number: ScaDS.AI. I also acknowledge financial support from the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under the National Research Data Infrastructure – NFDI 46/1 – 501864659 – NFDI4BioImage.

References

1. Loïc A. Royer. The future of bioimage analysis: a dialog between mind and machine. *Nature Methods*, 20(7):951–952, 2023. ISSN 1548-7105. doi: 10.1038/s41592-023-01930-y.
2. Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Scott Wen tau Yih, Daniel Fried, Sida Wang, and Tao Yu. Ds-1000: A natural and reliable benchmark for data science code generation, 2022.
3. Wei Lei, Cristina Fuster-Barceló, Georg Reder, et al. Bioimage.io chatbot: a community-driven ai assistant for integrative computational bioimaging. *Nature Methods*, 21:1368–1370, 2024. doi: 10.1038/s41592-024-02370-y.
4. Loïc A. Royer. Omega — harnessing the power of large language models for bioimage analysis. *Nature Methods*, 21(8):1371–1373, 2024. ISSN 1548-7105. doi: 10.1038/s41592-024-02310-w.

5. Robert Haase, Christian Tischer, Jean-Karim Hériché, and Nico Scherf. Benchmarking large language models for bio-image analysis code generation. *bioRxiv*, 2024. doi: 10.1101/2024.04.19.590278.
6. Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, et al. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021.
7. Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI Scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024.
8. Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues?, 2024.
9. GitHub. About github-hosted runners <https://docs.github.com/en/actions/using-github-hosted-runners/using-github-hosted-runners/about-github-hosted-runners>, 2024. Accessed: 2024-10-14.

Supplementary material

A)

[Plotting question]: Pair-plot #48

[Open](#) haesleinhuepf opened this issue last week · 1 comment

Assignees

haesleinhuepf commented last week

Owner

Simulate a dataset of 100 patients and three measurements. In a dataframe, the patients are rows and the measurements are columns. There is a fourth column where there is the stage classification (1, 2 or 3). Draw a pairplot of these measurements against each other and use the stage classification as hue.

[git-bob](#) try to do this.



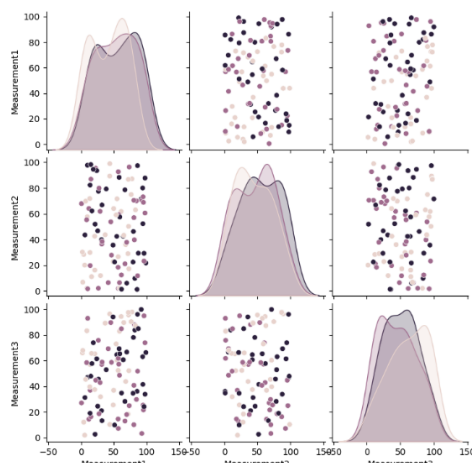
[haesleinhuepf](#) self-assigned this last week

github-actions bot commented last week

Contributor

This message was generated by [git-bob](#) (version: 0.11.0, model: gpt-4o-2024-08-06, [log](#)), an experimental AI-based assistant. It can make mistakes and has [limitations](#). Check its messages carefully.

The changes made to the haesleinhuepf/git-bob-playground repository addressed Issue #48 by creating a Jupyter Notebook, which simulates a dataset of 100 patients with three measurements and a stage classification. The notebook generates a pair-plot of these measurements with the stage classification as the hue. Additionally, the pair-plot created by the notebook is saved as an image file.



• An image of the pair-plot showing the simulated data with stage classification as hue was added.

• [pairplot_notebook.ipynb](#) A Jupyter Notebook was created to simulate patient data and generate a pair-plot.



B)

[git-bob-playground](#) / [pairplot_notebook.ipynb](#)

github-actions[bot] A Jupyter Notebook for si... dafa908 · last week

227 lines (227 loc) · 231 KB

Preview Code Blame

Raw Copy Download Edit

Pair-Plot of Simulated Patient Data

In this notebook, we'll simulate a dataset of 100 patients with three measurements and a stage classification. We'll then create a pair-plot to visualize these measurements with their stage classifications as the hue.

Step 1: Import Libraries

...

Step 2: Simulate the Dataset

We'll create a DataFrame with 100 patients, with three random measurements, and randomly assigned stages (1, 2, or 3).

In [2]:

```
# Seed for reproducibility
np.random.seed(0)

# Simulate measurements
measurements = np.random.rand(100, 3) * 100 # Example range from 0

# Simulate stage classification
stages = np.random.choice([1, 2, 3], size=100)

# Create DataFrame
df = pd.DataFrame(measurements, columns=['Measurement1', 'Measurement2', 'Measurement3'])
df['Stage'] = stages
```

...

Step 3: Create the Pair-Plot

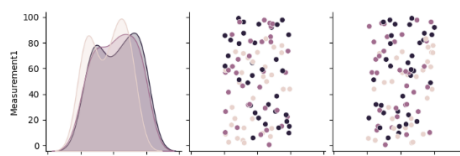
Now, we'll create a pair-plot using Seaborn, setting the `hue` parameter to the stage classification.

In [3]:

```
# Create the pairplot
pairplot = sns.pairplot(df, hue='Stage')

# Save the pairplot
pairplot.savefig('pairplot.png')

# Show the plot
plt.show()
```



...

Fig. S1. Use-case example for generating data analysis code: The user explains a scenario (A) and triggers git-bob (underlined in magenta). The AI-assistant generates and executes code and visualizes the resulting plot. The user can click on the link to the generated notebook (underlined in green) to go to the notebook (B) and read the code and see intermediate results. The shown notebook is an excerpt as indicated by "...". The entire discussion and corresponding code can be read online: <https://github.com/haesleinhuepf/git-bob-playground/issues/48>

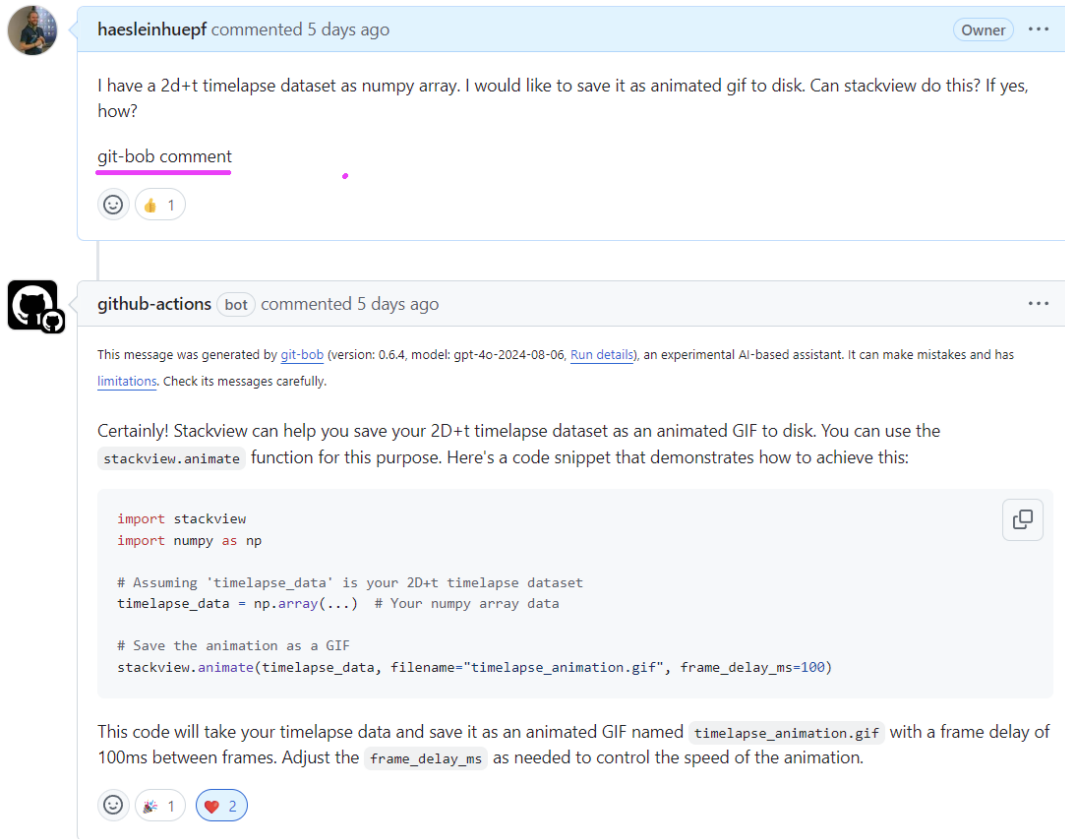


Fig. S2. Use-case example for supporting users: The assistant can be configured to act as expert on a specific Python library and answer user questions. Words triggering git-bob are underlined in magenta. The entire discussion and corresponding code can be read online: <https://github.com/haesleinhuepf/stackview/issues/79>

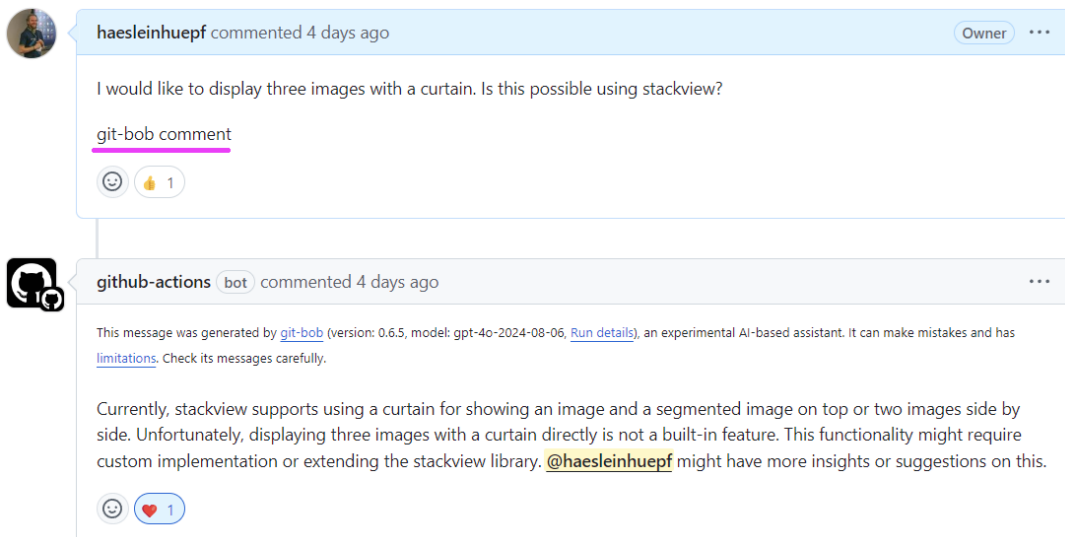
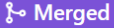



Fig. S3. Use-case example for asking an expert: The answer to the question shown here is "No", but this is nowhere written in the documentation or the configuration of the assistant. In this case the assistant is not sure, and it can be configured to forward a question to a maintainer of the library where the question arrived. Words triggering git-bob are underlined in magenta. The entire discussion and corresponding code can be read online: <https://github.com/haesleinhuepf/stackview/issues/80>

A)

Add numpy-style docstrings to _github_utilities.py #29


 Merged haesleinhuepf merged 1 commit into `main` from `mod-ajh4CjX8wR` on Aug 2


Conversation 0 Commits 1 Checks 0 Files changed 1


 **github-actions** bot commented on Aug 2 Contributor ...

This comment was generated by [git-bob](#), an AI-based assistant.

Added numpy-style docstrings for all functions in `src/git_bob/_github_utilities.py`. Closes [#22](#).







 Update file content d123ac1


 **haesleinhuepf** merged commit `daf0ec` into `main` on Aug 2 Revert

B)

Conversation 0 Commits 1 Checks 0 Files changed 1

Changes from all commits File filter Conversations Jump to  Ask Copilot Review

 78  `src/git_bob/_github_utilities.py` 

☐ Viewed  ...

		@@ -6,6 +6,18 @@
6	6	from github import Github
7	7	
8	8	def add_comment_to_issue(repository, issue, comment):
9	+	"""
10	+	Add a comment to a specific GitHub issue.
11	+	
12	+	Parameters
13	+	-----
14	+	repository : str
15	+	The full name of the GitHub repository (e.g., "username/repo-name").
16	+	issue : int
17	+	The issue number to add a comment to.
18	+	comment : str
19	+	The comment text to add to the issue.
20	+	"""
9	21	print(f"-> add_comment_to_issue({repository}, {issue}, ...)")
10	22	
11	23	access_token = os.getenv('GITHUB_API_KEY')

Fig. S4. Use-case example for adding and revising documentation in code: git-bob was used to partially write the code documentation of its own code. When asked to add documentation in a specific format, it sent a pull-request (A) and the human could inspect the code modifications (B, excerpt) before merging the code into the project's code base. The entire discussion and corresponding code can be read online: <https://github.com/haesleinhuepf/git-bob/pull/29>



haesleinhuepf commented 3 minutes ago

Member ...

In the folder "download_statistics", there are multiple files named for example "20240628.csv". The filename without ending is obviously a date, the creation date of the data. Write a jupyter notebook that opens all these files using pandas and plots over time how many records are in these files. Save the notebook in the notebooks directory and make sure the relative link the download_statistics folder is correct.

git-bob try again



1



github-actions bot commented 2 minutes ago

Contributor ...

This message was generated by [git-bob](#) (version: 0.8.2, model: github_models:gpt-4o, [log](#)), an experimental AI-based assistant. It can make mistakes and has [limitations](#). Check its messages carefully.

In this update, a Jupyter notebook was created to read CSV files and plot the number of records over time, resulting in the creation of [notebooks/plot_zenodo_links_over_time.ipynb](#). Additionally, an image file was generated:

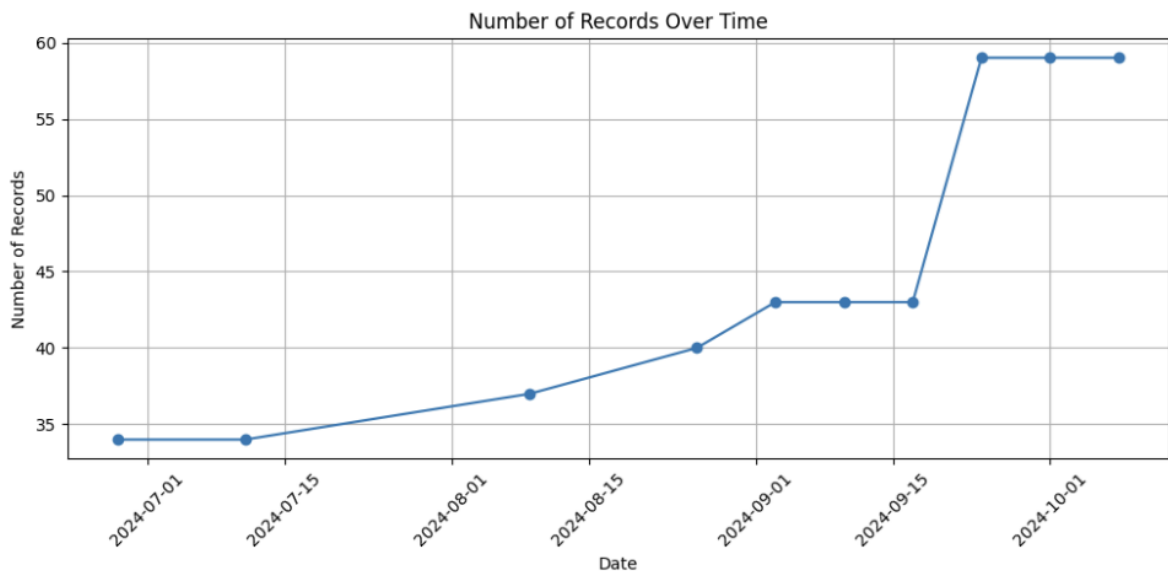


Fig. S5. Use-case example for plotting data: after explaining the folder structure of the project, it generates code for parsing a folder of CSV files and plotting results. Words triggering git-bob are underlined in magenta. The entire discussion and corresponding code can be read online: <https://github.com/NFDI4BIOIMAGE/training/issues/250>

A)

Add abstract #8

 Open haesleinhuepf opened this issue 5 minutes ago · 2 comments

haesleinhuepf commented 5 minutes ago

We need an abstract for the manuscript in the git-bob-manuscript.tex file. The manuscript will be submitted to a journal where life scientists are the target audience. I'd like to have an abstract focusing on the scientific challenges (transparency and knowledge exchange) when using LLMs. git-bob should be mentioned in the second half of the abstract.

git-bob think about this and answer in plain text with no markdown or tex styling.

github-actions (bot) commented 4 minutes ago

This message was generated by [git-bob](#) (version: 0.9.0, model: gpt-4o-2024-08-06, [log](#)), an experimental AI-based assistant. It can make mistakes and has [limitations](#). Check its messages carefully.

An abstract focusing on the scientific challenges of using LLMs in life sciences is provided below:

The integration of Large Language Models (LLMs) in scientific research presents both opportunities and challenges for life scientists. Key challenges include ensuring transparency in AI-generated content and facilitating efficient knowledge exchange among researchers. These issues arise from the opaque nature of AI-driven code generation and the informal sharing of AI insights, which may hinder reproducibility and collaboration. This paper introduces git-bob, an innovative AI-assistant designed to address these challenges by fostering an interactive and transparent collaboration platform within GitHub. By enabling seamless dialogue between humans and AI, git-bob ensures that AI contributions are transparent and verifiable. Moreover, it supports collaborative knowledge exchange, enhancing the interdisciplinary dialogue necessary for cutting-edge life sciences research. The open-source nature of git-bob further promotes accessibility and customization, positioning it as a vital tool in employing LLMs responsibly and effectively within scientific communities.


haesleinhuepf commented 2 minutes ago Author

Awesome! Add this abstract to the manuscript!

git-bob implement this.


github-actions (bot) mentioned this issue 1 minute ago

Add abstract to manuscript highlighting key challenges and git-bob solution. #9

 Open

B)

Add abstract to manuscript highlighting key challenges and git-bob solution. #9

 Open

github-actions wants to merge 1 commit into [main](#) from [git-bob-mod-FkM40ZPtHW](#)


Conversation 0 Commits 1 Checks 0 Files changed 1

github-actions (bot) commented 2 minutes ago

This message was generated by [git-bob](#) (version: 0.9.0, model: gpt-4o-2024-08-06, [log](#)), an experimental AI-based assistant. It can make mistakes and has [limitations](#). Check its messages carefully.

Added an abstract focusing on the scientific challenges of using LLMs in life sciences to the git-bob-manuscript.tex file, addressing issue [#8](#) by highlighting transparency, knowledge exchange, and introducing git-bob as a solution.

closes [#8](#)

 I added an abstract to the manuscript. 8b4a46a

C)

Conversation 0 Commits 1 Checks 0 Files changed 1

Changes from all commits File filter Conversations

Jump to 

 Ask Copilot Review

git-bob-manuscript.tex

Viewed

@@ -102,7 +102,7 @@
102 102
103 103 \begin{abstract}
104 104
105 - Abstract will be added later
105 + The integration of Large Language Models (LLMs) in scientific research presents both opportunities and challenges for life scientists. Key challenges include ensuring transparency in AI-generated content and facilitating efficient knowledge exchange among researchers. These issues arise from the opaque nature of AI-driven code generation and the informal sharing of AI insights, which may hinder reproducibility and collaboration. This paper introduces git-bob, an innovative AI-assistant designed to address these challenges by fostering an interactive and transparent collaboration platform within GitHub. By enabling seamless dialogue between humans and AI, git-bob ensures that AI contributions are transparent and verifiable. Moreover, it supports collaborative knowledge exchange, enhancing the interdisciplinary dialogue necessary for cutting-edge life sciences research. The open-source nature of git-bob further promotes accessibility and customization, positioning it as a vital tool in employing LLMs responsibly and effectively within scientific communities.
106 106
107 107 \end{abstract}
108 108

Fig. S6. Use-case example for working on scientific manuscripts: after a first draft of the manuscript was written, git-bob was asked to formulate an abstract (A). The abstract was then submitted as pull-request with a short description (B). The human can also review and potentially modify the proposed text in this online interface (C). Words triggering git-bob are underlined in magenta. The entire discussion can be read online: <https://github.com/haesleinhuepf/git-bob-manuscript/issues/8> and <https://github.com/haesleinhuepf/git-bob-manuscript/pull/9>.