A Platform for the Biomedical Application of Large Language Models

This manuscript (permalink) was automatically generated from biocypher/biochatter-paper@f083f32 on December 29, 2023.

Authors

- **Sebastian Lobentanzer** ✉
  0000-0003-3399-6695 · slobentanzer · slobentanzer
  Heidelberg University, Faculty of Medicine and Heidelberg University Hospital, Institute for Computational Biomedicine, Heidelberg, Germany

- **Andreas Maier**
  0000-0003-4408-0068 · andimajore
  Institute for Computational Systems Biology, University of Hamburg, Hamburg, Germany

- **Julio Saez-Rodriguez** ✉
  0000-0002-8552-8976 · saezrodriguez · saezlab
  Heidelberg University, Faculty of Medicine and Heidelberg University Hospital, Institute for Computational Biomedicine, Heidelberg, Germany

✉ — Correspondence possible via GitHub Issues or email to Sebastian Lobentanzer <sebastian.lobentanzer@gmail.com>, Julio Saez-Rodriguez <pub.saez@uni-heidelberg.de>.

Abstract

Current-generation Large Language Models (LLMs) have stirred enormous interest in the recent months, yielding great potential for accessibility and automation, while simultaneously posing significant challenges and risk of misuse. To facilitate interfacing with LLMs in the biomedical space, while at the same time safeguarding their functionalities through sensible constraints, we propose a dedicated, open-source framework: BioChatter. Based on open-source software packages, we synergise the many functionalities that currently develop around LLMs, such as knowledge integration and retrieval augmented generation, model chaining, and benchmarks, resulting in an easy-to-use and inclusive framework for application in many use cases of biomedical informatics. We focus on robust and user-friendly implementation, including ways to deploy privacy-preserving local open-source LLMs. We demonstrate use cases via a multi-purpose web app, ChatGSE (https://chat.biocypher.org), and provide documentation, support, and an open community to all interested researchers.

Introduction

Despite our technological advances, biology and biomedicine continue to pose incredible challenges [1,2]. We measure more and more data points with ever-increasing resolution to such a degree that their analysis and interpretation have become the bottleneck for their exploitation [2]. One reason for this challenge may be the inherent limitation of human knowledge [3]: Even seasoned domain experts cannot know the implications of every molecule, be it metabolite, DNA, RNA, or protein, even in their own domain. In addition, biological events are context-dependent, for instance with respect to a cell type or specific disease.

Large Language Models (LLMs) of the current generation, on the other hand, can access enormous amounts of knowledge, encoded (incomprehensibly) in their billions of parameters [4,5,6]. Trained correctly, they can recall and combine virtually limitless knowledge from their training set. ChatGPT has taken the world by storm, and many biomedical researchers already use LLMs in their daily work, for general as well as bioinformatics-specific tasks [7,8]. However, the current, predominantly manual, way of interacting with LLMs is virtually non-reproducible, and their behaviour can be erratic. For instance, they are known to confabulate: they make up facts as they go along, and, to make matters worse, are convinced - and convincing - regarding the truth of their confabulations [8,9]. While current efforts towards Artificial General Intelligence manage to ameliorate some of the shortcomings by ensembling multiple models [10] with long-term memory stores [11], the current generation of AI does not inspire adequate trust to be applied to biomedical problems without supervision [9]. Additionally, biomedicine demands greater care in data privacy, licensing, and transparency than most other real-world issues.

Computational biomedicine involves many tasks that could be assisted by LLMs, such as the interpretation of experimental results, the design of experiments, the evaluation of literature, and the exploration of web resources. To improve and accelerate these tasks, we have developed BioChatter, a platform for communicating with LLMs specifically tuned to biomedical research (Figure 1). The platform guides the human researcher intuitively through the interaction with the model, while counteracting the problematic behaviours of the LLM. Since the interaction is mainly based on plain text (in any language), it can be used by virtually any researcher.

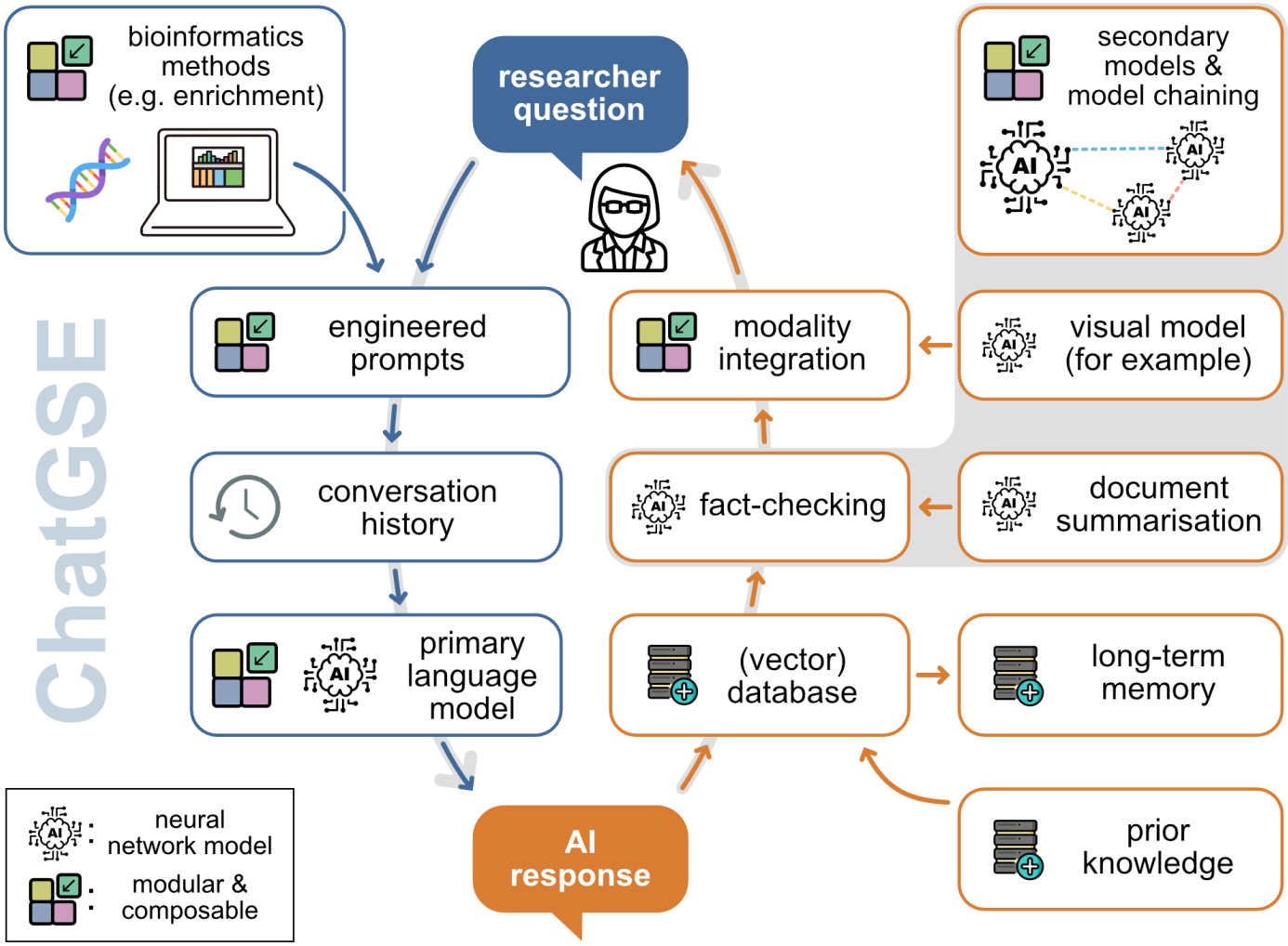

Figure 1: The BioChatter composable platform architecture (simplified). Many questions arise in daily biomedical research practice, for instance, interpretation of experimental results or the use of a web resource (top left). BioChatter's main response circuit (blue) composes a number of specifically engineered prompts and passes them (and a conversation history) to the primary LLM, which generates a response for the user based on all inputs. This response is simultaneously used to prompt the secondary circuit (orange), which fulfils auxiliary tasks to complement the primary response. In particular, using search, the secondary circuit queries a database as prior knowledge repository and compares annotations to the primary response. The knowledge graph can also serve as long-term memory extension of the model. Further, an independent LLM receives the primary response for fact-checking, which can be supplemented with context-specific information by a retrieval augmented generation process. If this "second opinion" differs from the primary response, a warning is issued. The platform is composable in all aspects, allowing arbitrary extensions to other, specialised models for additional tasks orchestrated by the primary LLM.

Results

BioChatter (https://github.com/biocypher/biochatter) is a python framework that provides an easy-to-use interface to interact with LLMs and auxiliary technologies via an intuitive API (application programming interface). This way, its functionality can be integrated into any number of user interfaces, such as web apps, command line interfaces, or Jupyter notebooks. The framework is designed to be composable, meaning that any of its components can be exchanged with other implementations, for instance, to use a different LLM or knowledge graph (KG). Functionalities include:

- **basic question answering** with LLMs hosted by providers (such as OpenAI) as well as locally deployed open-source models
- **reproducible prompt engineering** to guide the LLM towards a specific task or behaviour

- **benchmarking** of LLMs, prompts, and other components
- **knowledge graph querying** with automatic integration of any KG created in the BioCypher framework [12]
- **retrieval augmented generation** (RAG) using vector database embeddings of user-provided literature
- **model chaining** to orchestrate multiple LLMs and other models in a single conversation using the LangChain framework [10]
- **fact-checking** of LLM responses using a second LLM

In the following, we briefly describe these components, which are demonstrated in the ChatGSE web app (https://chat.biocypher.org).

Question Answering and LLM Connectivity

The core functionality of BioChatter is to interact with LLMs. The framework supports both leading proprietary models such as the GPT series from OpenAI as well as open-source models such as LLaMA2 [13] via a flexible open-source deployment framework [14] (see Methods). Currently, the most powerful conversational AI platform, ChatGPT (OpenAI), is surrounded by data privacy concerns [15]. We address this issue in two ways. Firstly, we provide access to the different OpenAI models through their API, which is subject to different, more stringent data protection than the web interface [16], most importantly by disallowing reuse of user inputs for subsequent model training. Secondly, we aim to preferentially support open-source LLMs to facilitate more transparency in their application and increase data privacy by being able to run a model locally on dedicated hardware and end-user devices [17]. By building on LangChain [10], we support dozens of LLM providers, such as the Xorbits Inference and Hugging Face APIs [14], which can be used to query any of the more than 100 000 open-source models on Hugging Face Hub [18], for instance those on its LLM leaderboard [19]. Although OpenAI's models currently vastly outperform any alternatives in terms of both LLM performance and API convenience, we expect many open-source developments in this area in the future [20]. Therefore, we support plug-and-play exchange of models to enhance biomedical AI-readiness, and we implement a bespoke benchmarking framework for the biomedical application of LLMs.

Prompt Engineering

An essential property of LLMs is their sensitivity to the prompt, i.e., the initial input that guides the model towards a specific task or behaviour. Prompt engineering is an emerging discipline of practical AI, and as such there are no established best practices [21]. Current approaches are mostly trial-and-error-based manual engineering, which is not reproducible and changes with every new model [20]. To address this issue, we include a prompt engineering framework in BioChatter that allows the preservation of prompt sets for specific tasks, which can be shared and reused by the community. In addition, to facilitate the scaling of prompt engineering, we integrate this framework in the benchmarking pipeline, which allows the automated evaluation of prompt sets as new models are published.

Benchmarking

To facilitate the reproducible evaluation of LLMs, we implement a benchmarking framework that allows the comparison of models, prompt sets, and other components of the pipeline. Built on the generic Pytest framework [22], it allows the automated evaluation of a matrix of all possible

combinations of components. The results are stored and displayed on our website for easy comparison, and the benchmark is updated upon the release of new models and extensions to the datasets. We create a bespoke biomedical benchmark for multiple reasons: Firstly, the biomedical domain has its own tasks and requirements, and creating a bespoke benchmark allows us to be more precise in the evaluation of components [20]. Secondly, we aim to create benchmark datasets that are complementary to the existing, general purpose benchmarks and leaderboards for LLMs [19]. Thirdly, we aim to prevent leakage of the benchmark data into the training data of the models, which is a known issue in the general purpose benchmarks, also called memorisation or contamination [23]. To achieve this goal, we implemented an encrypted pipeline that contains the benchmark datasets and is only accessible to the workflow that executes the benchmark (see Methods).

Figure?

Knowledge Graphs

Knowledge graphs (KGs) are a powerful tool to represent and query knowledge in a structured manner. With BioCypher [12], we have developed a framework to create KGs from biomedical data in a user-friendly manner while also semantically grounding the data in ontologies. BioChatter is an extension of the BioCypher ecosystem, elevating its user-friendliness further by allowing natural language interactions with the data; any BioCypher KG is automatically compatible with BioChatter. We use information generated in the build process of BioCypher KGs to tune BioChatter's understanding of the data structures and contents, thereby increasing the efficiency of LLM-based KG querying (see Methods). In addition, the ability to connect to any BioCypher KG allows the integration of prior knowledge into the LLM's reasoning, which can be used to ground the model's responses in the context of the KG via in-context learning / RAG (see below).

Retrieval Augmented Generation

LLM confabulation is a major issue for biomedical applications, where the consequences of incorrect information can be severe. One popular way of addressing this issue is to apply "in-context learning," which is also more recently referred to as "retrieval augmented generation" (RAG) [24]. While this can be done by processing structured knowledge, for instance from KGs, it is often more efficient to use a semantic search engine to retrieve relevant information from unstructured data sources such as literature. To this end, we allow the management and integration of vector databases in the BioChatter framework. The user is able to connect to a vector database, embed an arbitrary number of documents, and then use semantic search to improve the model prompts by adding text fragments relevant to the given question (see Methods).

Model Chaining and Fact Checking

LLMs cannot only seamlessly interact with human users, but also with other LLMs as well as many other types of models. They understand API calls and can therefore theoretically orchestrate complex multi-step tasks [25,26]. However, implementation is not trivial and the complex process can lead to unpredictable behaviours. We aim to improve the stability of model chaining in biomedical applications by developing bespoke approaches for common biomedical tasks, such as interpretation and design of experiments, evaluating literature, and exploring web resources. While we focus on reusing existing open-source frameworks such as LangChain [10], we also develop bespoke solutions where necessary to provide stability for the given application. As an example, we implement a fact-checking module that uses a second LLM to evaluate the factual correctness of the primary LLM's responses continuously during the conversation (see Methods).

Discussion

The fast pace of developments around current-generation LLMs poses a particular challenge to the biomedical community. While the potential of these models is enormous, their application is not straightforward, and their use requires a certain level of expertise. In addition, biomedical research is often performed in a siloed way due to the complexity of the domain and systemic incentives that work against open science and collaboration [27]. Inspired by the productivity of open source libraries such as LangChain [10], we propose an open framework that allows biomedical researchers to focus on the application of LLMs as opposed to engineering challenges. To keep the framework effective and sustainable, we focus on reusing existing open-source libraries and tools, while adapting the advancements from the wider LLM community to the biomedical domain.

To account for the requirements of biomedical research workflows, we take particular care to guarantee robustness and objective evaluation of LLM behaviour and their performance in interaction with other parts of the framework. We achieve this goal by implementing a living benchmarking framework that allows the automated evaluation of LLMs, prompts, and other components. As even the most recent and biomedicine-specific benchmarking efforts are small-scale manual approaches that do not consider the full matrix of possible combinations of components [20], such a framework is a necessary step towards the objective evaluation of LLMs. We prevent data leakage from the benchmark datasets into the training data of new models by encryption, which is essential for the sustainability of the benchmark as new models are released. The living benchmark will be updated with new questions and tasks as they arise in the community [20].

We facilitate access to LLMs by allowing the use of both proprietary and open-source models, and we provide a flexible deployment framework for the latter. Proprietary models are currently the most economic solution for accessing state-of-the-art models, and as such primarily suited for users just starting out or lacking the resources to deploy their own models. In contrast, open-source models are quickly catching up in terms of performance [20], and they are essential for the sustainability of the field. We allow self-hosting of open-source models on any scale, from dedicated hardware with GPUs, to local deployment on end-user laptops, to browser-based deployment using web technology.

Limitations

Depending on generic open-source libraries such as LangChain [10] and Pytest [22] allows us to focus on the biomedical domain but also introduces dependencies on these libraries. While we support those upstream libraries via pull requests, we depend on their maintainers for future updates. In addition, keeping up with these rapid developments is demanding on developer time, which is only sustainable in a community-driven open-source effort.

Most importantly, the current generation of LLMs is not yet ready for unsupervised use in biomedical research. While we have taken steps to mitigate the risks of using LLMs, such as independent benchmarks, fact-checking, and knowledge graph querying, we cannot guarantee that the models will not produce harmful outputs. We see current LLMs, particularly in the scope of the BioCypher ecosystem, as helpful tools to assist human researchers, alleviating menial and repetitive tasks and helping with technical aspects such as query languages. They are not meant to replace human ingenuity and expertise, but to augment it with their complemetary strengths.

Future directions

Autonomous agents for trivial tasks have already been developed on the basis of LLMs, and we expect this field to mature in the future [26]. As research on agent behaviour progresses, we will integrate these developments into the BioChatter framework to allow the creation of helpful assistants for

biomedical research. All framework developments will be performed in light of the ethical implications of LLMs, and we will continue to support the use of open-source models to increase transparency and data privacy. While we focus on the biomedical field, the concept of our frameworks can easily be extended to other scientific domains by adjusting domain-specific prompts and data inputs, which are accessible in a composable and user-friendly manner in our frameworks [12].

Our Python library is developed openly on GitHub (https://github.com/biocypher/biochatter) and can be integrated into any number of user interface solutions. We develop under the permissive MIT licence and encourage contributions and suggestions from the community with regard to the addition of bioinformatics tool integrations, prompt engineering, benchmarking, and any other feature.

Methods

BioChatter is a Python library, currently supporting Python 3.10-3.12, and generally the three most recent releases, which we ensure with a continuous integration pipeline on GitHub. ChatGSE is a web app based on the Streamlit framework (version 1.21.0, https://streamlit.io), which is written in Python and can be deployed locally or on a server (https://github.com/biocypher/ChatGSE). It is mainly used for demonstrating the various applications of the BioChatter framework and API. For an up-to-date overview and preview of current functionality of the platform, please visit the online preview. ChatGSE Next (https://github.com/biocypher/chatgse-next) is a modern web app with server-client architecture, based on the open-source template of ChatGPT-Next-Web (https://github.com/ChatGPTNextWeb/ChatGPT-Next-Web). It is written in TypeScript using Flask and Node.js and demonstrates the use of BioChatter in a modern web app. All packages are developed openly and according to modern standards of software development [28]; we use the permissive MIT licence to encourage downstream use and development. We include a code of conduct and contributor guidelines to offer accessibility and inclusivity to all that are interested in contributing to the framework.

Benchmarking

The benchmarking framework implements a matrix of component combinations using the parameterisation feature of Pytest [22]. This allows the automated evaluation of all possible combinations of components, such as LLMs, prompts, and datasets. The results are stored in a database and displayed on the website for easy comparison. The benchmark is updated upon the release of new models and extensions to the datasets. The individual dimensions of the matrix are:

- **LLMs**: Testing proprietary (OpenAI) and open-source models (commonly using the Xorbits Inference API and HuggingFace models) against the same set of tasks is the primary aim of our benchmarking framework. We facilitate the automation of testing by including a programmatic way of deploying open-source models.

- **prompts**: Since model performance can rely on prompts dramatically, a set of prompts for each task with varying degrees of specificity and fixed as well as variable components is used to evaluate this variability.

- **datasets**: We test various tasks using a set of datasets for each task in question-answer-style.

- **data processing**: Some data processing steps can have great impact on the downstream performance of LLMs. For instance, we test the conversion of numbers (which LLMs are notoriously bad at handling) to categorical text (e.g., low, medium, high).

- **model quantisations**: We test a set of quantisations for each model (where available) to account for the trade-off between model size and performance.

- **model parameters**: Where suitable, we test a set of parameters for each model, such as "temperature," which determines the reproducibility of model responses.

- **integrations**: We write dedicated tests for specific tasks that require integrations, for instance with knowledge graphs or vector databases.

- **stochasticity**: To account for variablity in model responses, we include a parameter to run each test multiple times and generate summary statistics.

The Pytest framework is implemented at https://github.com/biocypher/biochatter/blob/main/benchmark, and more information and results are available at https://biocypher.github.io/biochatter/benchmark.

To prevent leakage of benchmarking data (and subsequent contamination of future LLMs), we implement an encryption routine on the benchmark datasets. The encryption is performed using a hybrid encryption scheme, where the data is encrypted with a symmetric key, which is in turn encrypted with an asymmetric key. The datasets are stored in a dedicated encrypted pipeline that is only accessible to the workflow that executes the benchmark. These processes are implemented at https://github.com/biocypher/llm-test-dataset and accessed from the benchmark procedure in BioChatter.

Knowledge Graphs

We utilise the close connection between BioChatter and the BioCypher framework [12] to integrate knowledge graph (KG) queries into the BioChatter API. In the BioCypher KG creation, we use a configuration file to map KG contents to ontology terms, including information about each of the entities. For instance, we detail the properties of a node and the source and target classes of an edge. Additionally, during the KG build process, we enrich this information and save it to a YAML file and, optionally, directly to the KG. This information is used by BioChatter to tune its understanding of the KG, which allows the LLM to query the KG more efficiently. By understanding the context of the KG, the exact contents, and the exact spelling of all identifiers and properties, we effectively support the LLM in generating correct queries. To illustrate the usage of this feature, we provide a demonstration repository at https://github.com/biocypher/pole including a KG build procedure and ChatGSE app, which can be run using a single Docker Compose command.

Retrieval Augmented Generation

While the general knowledge of current LLMs is extensive, they may not know how to prioritise very specific scientific results, or they may not have had access to some research articles in their training data (e.g., due to their recency or licensing issues). To bridge this gap, we can provide additional information from relevant publications to the model via the prompt. However, we cannot add entire publications to the prompt, since the input length of current models still is restricted; we need to isolate the information that is specifically relevant to the question given by the user. To find this information, we perform a semantic similarity search between the user's question and the contents of user-provided scientific articles (or other texts). The most efficient way to do this mapping is by using a vector database.

The contextual background information provided by the user (e.g., by uploading a scientific article of prior work related to the experiment to be interpreted) is split into pieces suitable to be digested by

the LLM, which are individually embedded by the model. These embeddings (represented by vectors) are used to store the text fragments in a vector database; the storage as vectors allows fast and efficient retrieval of similar entities via the comparison of individual vectors. For example, the two sentences “Amyloid beta levels are associated with Alzheimer’s Disease stage.” and “One of the most important clinical markers of AD progression is the amount of deposited A-beta 42.” would be closely associated in a vector database (given the embedding model is of sufficient quality, i.e., similar to GPT-3 or better), while traditional text-based similarity metrics probably would not identify them as highly similar.

By comparing the user’s question to prior knowledge in the vector database, we can extract the relevant pieces of information from the entire background. Even better, we can first use an LLM to generate an answer to the user’s question and then use this answer to query the vector database for relevant information. Regardless of whether the initial answer is correct, it is likely that the “fake answer” is more semantically similar to the relevant pieces of information than the user’s question [29]. Semantic search results (for instance, single sentences directly related to the topic of the question) are then sufficiently small to be added to the prompt. In this way, the model can learn from additional context without the need for retraining or fine-tuning. This method is sometimes described as in-context learning [24] or retrieval-augmented generation [30].

To provide access to this functionality in BioChatter, we implement classes for the connection to, and management of, vector database systems (in the vectorstore_host.py module), and for performing semantic search on the vector database and injecting the results into the prompt (in the vectorstore.py module). To demonstrate the use of the API, we add a “Retrieval Augmented Generation” tab to the ChatGSE preview app that allows the upload of text documents to be added to a vector database, which then can be queried to add contextual information to the prompt sent to the primary model. This contextual information is transparently displayed. Since this functionality requires a connection to a vector database system, we provide connectivity to a Milvus server, including a way to start the server in conjunction with a BioCypher knowledge graph and the ChatGSE app in one Docker Compose workflow.

Deployment of Open-Source Models

To facilitate access to open-source models, we adopt a flexible deployment framework based on the Xorbits Inference API [14]. We provide Docker images for the automatic deployment of the API and the models. Xorbits Inference includes a large number of open-source models out of the box, and new models from Hugging Face Hub [31] can be added using the intuitive graphical user interface.

In addition, we provide fully browser-based deployment of LLMs using WebAssembly (WASM) and the web-llm library (https://github.com/mlc-ai/web-llm). We host a local model using web-llm, which is accessed by the BioChatter server within ChatGSE Next. This combination creates a secure conversational environment and minimises the risks associated with data breaches. Hosting local LLMs using WebLLM allows them to run without an internet connection in a WASM module. This architecture thus enables complete data security and is limited only by the resources of the host computer. While the same can be achieved with a local deployment of the Xorbits Inference API, the browser-based deployment is more user-friendly and does not require any additional software.

Model Chaining

The ability of LLMs to control external software, including other LLMs, opens up a wide range of possibilities for the orchestration of complex tasks. A simple example is the implementation of a correcting agent, which receives the output of the primary model and checks it for factual correctness. If the agent detects an error, it can prompt the primary model to correct its output, or forward this

correction to the user directly. Since this relies on the internal knowledge base of the correcting agent, the same caveats apply, as the correcting agent may confabulate as well. However, since the agent is independent of the primary model (being set up with dedicated prompts), it is less likely to confabulate in the same way.

This approach can be extended to a more complex model chain, where the correcting agent, for example, can query a knowledge graph or a vector database to ground its responses in prior knowledge. These chains are easy to implement, and some are available out of the box in the LangChain framework [10]. However, they can behave unpredictably, which increases with the number of links in the chain, and as such should be tightly controlled. They also add to the computational burden of the system, which is particularly relevant for deployments on end-user devices.

Author Contributions

SL conceptualised and developed the platform and wrote the manuscript. AM implemented the local deployment functionality. JSR supervised the project, revised the manuscript, and acquired funding. All authors read and approved the final manuscript.

Acknowledgements

We thank Hanna Schumacher, Daniel Dimitrov, Pau Badia i Mompel, and Aurelien Dugourd for feedback on the original draft of the manuscript and the software.

Conflict of Interest

JSR reports funding from GSK, Pfizer and Sanofi and fees from Travere Therapeutics, Stadapharm and Astex Pharmaceuticals.

References

1. **Study reveals cancer's 'infinite' ability to evolve**
James Gallagher
BBC News (2023-04-12) https://www.bbc.com/news/health-65252510

2. **Current progress and open challenges for applying deep learning across the biosciences**
Nicolae Sapoval, Amirali Aghazadeh, Michael G Nute, Dinler A Antunes, Advait Balaji, Richard Baraniuk, CJ Barberan, Ruth Dannenfelser, Chen Dun, Mohammadamin Edrisi, … Todd J Treangen
Nature Communications (2022-04-01) https://doi.org/gp26xk
DOI: 10.1038/s41467-022-29268-7 · PMID: 35365602 · PMCID: PMC8976012

3. **Capacity limits of information processing in the brain**
René Marois, Jason Ivanoff
Trends in Cognitive Sciences (2005-06) https://doi.org/d5gmqt
DOI: 10.1016/j.tics.2005.04.010 · PMID: 15925809

4. **PaLM: Scaling Language Modeling with Pathways**
Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, … Noah Fiedel
arXiv (2022) https://doi.org/kfxf
DOI: 10.48550/arxiv.2204.02311

5. **LaMDA: Language Models for Dialog Applications**
Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, … Quoc Le
arXiv (2022) https://doi.org/kmfc
DOI: 10.48550/arxiv.2201.08239

6. **GPT-4 Technical Report**
OpenAI, :, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, … Barret Zoph
arXiv (2023) https://doi.org/grx4cb
DOI: 10.48550/arxiv.2303.08774

7. **Assessing GPT-4 for cell type annotation in single-cell RNA-seq analysis**
Wenpin Hou, Zhicheng Ji
Cold Spring Harbor Laboratory (2023-04-21) https://doi.org/gsznzg
DOI: 10.1101/2023.04.16.537094 · PMID: 37131626 · PMCID: PMC10153208

8. **How will generative AI disrupt data science in drug discovery?**
Jean-Philippe Vert
Nature Biotechnology (2023-05-08) https://doi.org/gsznzd
DOI: 10.1038/s41587-023-01789-6 · PMID: 37156917

9. **Foundation models for generalist medical artificial intelligence**
Michael Moor, Oishi Banerjee, Zahra Shakeri Hossein Abad, Harlan M Krumholz, Jure Leskovec, Eric J Topol, Pranav Rajpurkar
Nature (2023-04-12) https://doi.org/gr4td4
DOI: 10.1038/s41586-023-05881-4 · PMID: 37045921

10. □□ **Langchain** https://python.langchain.com/

11. **AutoGPT Official**
AutoGPT Official
(2023-12-18) https://autogpt.net/

12. **Democratizing knowledge representation with BioCypher**
Sebastian Lobentanzer, Patrick Aloy, Jan Baumbach, Balazs Bohar, Vincent J Carey, Pornpimol Charoentong, Katharina Danhauser, Tunca Doğan, Johann Dreo, Ian Dunham, … Julio Saez-Rodriguez
Nature Biotechnology (2023-06-19) https://doi.org/gszqjr
DOI: 10.1038/s41587-023-01848-y · PMID: 37337100

13. **Llama 2: Open Foundation and Fine-Tuned Chat Models**
Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, … Thomas Scialom
arXiv (2023) https://doi.org/ktkj
DOI: 10.48550/arxiv.2307.09288

14. **Xorbits Inference: Model Serving Made Easy 🤖**
Xorbits
(2023-12-29) https://github.com/xorbitsai/inference

15. https://www.reuters.com/technology/european-data-protection-board-discussing-ai-policy-thursday-meeting-2023-04-13/

16. **Terms of use** https://openai.com/policies/terms-of-use

17. **Why open-source generative AI models are an ethical way forward for science**
Arthur Spirling
Nature (2023-04-18) https://doi.org/gsqx6v
DOI: 10.1038/d41586-023-01295-4 · PMID: 37072520

18. **Hugging Face Hub documentation** https://huggingface.co/docs/hub/index

19. **Open LLM Leaderboard - a Hugging Face Space by HuggingFaceH4**
https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard

20. **BioLLMBench: A Comprehensive Benchmarking of Large Language Models in Bioinformatics**
Varuni Sarwal, Viorel Munteanu, Timur Suhodolschi, Dumitru Ciorba, Eleazar Eskin, Wei Wang, Serghei Mangul
Cold Spring Harbor Laboratory (2023-12-20) https://doi.org/gtbgvk
DOI: 10.1101/2023.12.19.572483

21. **A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT**
Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, Douglas C Schmidt
arXiv (2023) https://doi.org/grxct8
DOI: 10.48550/arxiv.2302.11382

22. **pytest-dev/pytest**
pytest-dev
(2023-12-29) https://github.com/pytest-dev/pytest

23. **NLP Evaluation in trouble: On the Need to Measure LLM Data Contamination for each Benchmark**

Oscar Sainz, Jon Ander Campos, Iker García-Ferrero, Julen Etxaniz, Oier Lopez de Lacalle, Eneko Agirre
arXiv (2023) https://doi.org/gtbgvp
DOI: 10.48550/arxiv.2310.18018

24. **HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in Hugging Face**
Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, Yueting Zhuang
arXiv (2023) https://doi.org/gskd97
DOI: 10.48550/arxiv.2303.17580

25. **Gorilla: Large Language Model Connected with Massive APIs**
Shishir G Patil, Tianjun Zhang, Xin Wang, Joseph E Gonzalez
arXiv (2023) https://doi.org/gtbgvm
DOI: 10.48550/arxiv.2305.15334

26. **A Survey on Large Language Model based Autonomous Agents**
Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, … Ji-Rong Wen
arXiv (2023) https://doi.org/gsv93m
DOI: 10.48550/arxiv.2308.11432

27. **Scientific Utopia**
Brian A Nosek, Jeffrey R Spies, Matt Motyl
Perspectives on Psychological Science (2012-11) https://doi.org/f4fc2k
DOI: 10.1177/1745691612459058 · PMID: 26168121 · PMCID: PMC10540222

28. **The TRUST Principles for digital repositories**
Dawei Lin, Jonathan Crabtree, Ingrid Dillo, Robert R Downs, Rorie Edmunds, David Giaretta, Marisa De Giusti, Hervé L'Hours, Wim Hugo, Reyna Jenkyns, … John Westbrook
Scientific Data (2020-05-14) https://doi.org/ggwrtj
DOI: 10.1038/s41597-020-0486-7 · PMID: 32409645 · PMCID: PMC7224370

29. **Large Language Models for Information Retrieval: A Survey**
Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Zhicheng Dou, Ji-Rong Wen
arXiv (2023) https://doi.org/gtbgvn
DOI: 10.48550/arxiv.2308.07107

30. **Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks**
Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, … Douwe Kiela
Advances in Neural Information Processing Systems (2020)
https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf

31. **Hugging Face – The AI community building the future.** (2023-12-13) https://huggingface.co/