

Article

Autonomous chemical research with large language models

<https://doi.org/10.1038/s41586-023-06792-0>

Daniil A. Boiko¹, Robert MacKnight¹, Ben Kline² & Gabe Gomes^{1,3,4}✉

Received: 20 April 2023

Accepted: 27 October 2023

Published online: 20 December 2023

Open access

 Check for updates

Transformer-based large language models are making significant strides in various fields, such as natural language processing^{1–5}, biology^{6,7}, chemistry^{8–10} and computer programming^{11,12}. Here, we show the development and capabilities of Coscientist, an artificial intelligence system driven by GPT-4 that autonomously designs, plans and performs complex experiments by incorporating large language models empowered by tools such as internet and documentation search, code execution and experimental automation. Coscientist showcases its potential for accelerating research across six diverse tasks, including the successful reaction optimization of palladium-catalysed cross-couplings, while exhibiting advanced capabilities for (semi-)autonomous experimental design and execution. Our findings demonstrate the versatility, efficacy and explainability of artificial intelligence systems like Coscientist in advancing research.

Large language models (LLMs), particularly transformer-based models, are experiencing rapid advancements in recent years. These models have been successfully applied to various domains, including natural language^{1–5}, biological^{6,7} and chemical research^{8–10} as well as code generation^{11,12}. Extreme scaling of models¹³, as demonstrated by OpenAI, has led to significant breakthroughs in the field^{1,14}. Moreover, techniques such as reinforcement learning from human feedback¹⁵ can considerably enhance the quality of generated text and the models' capability to perform diverse tasks while reasoning about their decisions¹⁶.

On 14 March 2023, OpenAI released their most capable LLM to date, GPT-4¹⁴. Although specific details about the model training, sizes and data used are limited in GPT-4's technical report, OpenAI researchers have provided substantial evidence of the model's exceptional problem-solving abilities. Those include—but are not limited to—high percentiles on the SAT and BAR examinations, LeetCode challenges and contextual explanations from images, including niche jokes¹⁴. Moreover, the technical report provides an example of how the model can be used to address chemistry-related problems.

Simultaneously, substantial progress has been made toward the automation of chemical research. Examples range from the autonomous discovery^{17,18} and optimization of organic reactions¹⁹ to the development of automated flow systems^{20,21} and mobile platforms²².

The combination of laboratory automation technologies with powerful LLMs opens the door to the development of a sought-after system that autonomously designs and executes scientific experiments. To accomplish this, we intended to address the following questions. What are the capabilities of LLMs in the scientific process? What degree of autonomy can we achieve? How can we understand the decisions made by autonomous agents?

In this work, we present a multi-LLMs-based intelligent agent (hereafter simply called Coscientist) capable of autonomous design, planning and performance of complex scientific experiments. Coscientist

can use tools to browse the internet and relevant documentation, use robotic experimentation application programming interfaces (APIs) and leverage other LLMs for various tasks. This work has been done independently and in parallel to other works on autonomous agents^{23–25}, with ChemCrow²⁶ serving as another example in the chemistry domain. In this paper, we demonstrate the versatility and performance of Coscientist in six tasks: (1) planning chemical syntheses of known compounds using publicly available data; (2) efficiently searching and navigating through extensive hardware documentation; (3) using documentation to execute high-level commands in a cloud laboratory; (4) precisely controlling liquid handling instruments with low-level instructions; (5) tackling complex scientific tasks that demand simultaneous use of multiple hardware modules and integration of diverse data sources; and (6) solving optimization problems requiring analyses of previously collected experimental data.

Coscientist system architecture

Coscientist acquires the necessary knowledge to solve a complex problem by interacting with multiple modules (web and documentation search, code execution) and by performing experiments. The main module ('Planner') has the goal of planning, based on the user input by invoking the commands defined below. The Planner is a GPT-4 chat completion instance serving the role of an assistant. The initial user input along with command outputs are treated as user messages to the Planner. System prompts (static inputs defining the LLMs' goals) for the Planner are engineered^{1,27} in a modular fashion, described as four commands that define the action space: 'GOOGLE', 'PYTHON', 'DOCUMENTATION' and 'EXPERIMENT'. The Planner calls on each of these commands as needed to collect knowledge. The GOOGLE command is responsible for searching the internet with the 'Web searcher' module, which is another LLM itself.

¹Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA, USA. ²Emerald Cloud Lab, South San Francisco, CA, USA. ³Department of Chemistry, Carnegie Mellon University, Pittsburgh, PA, USA. ⁴Wilton E. Scott Institute for Energy Innovation, Carnegie Mellon University, Pittsburgh, PA, USA. ✉e-mail: gabegomes@cmu.edu

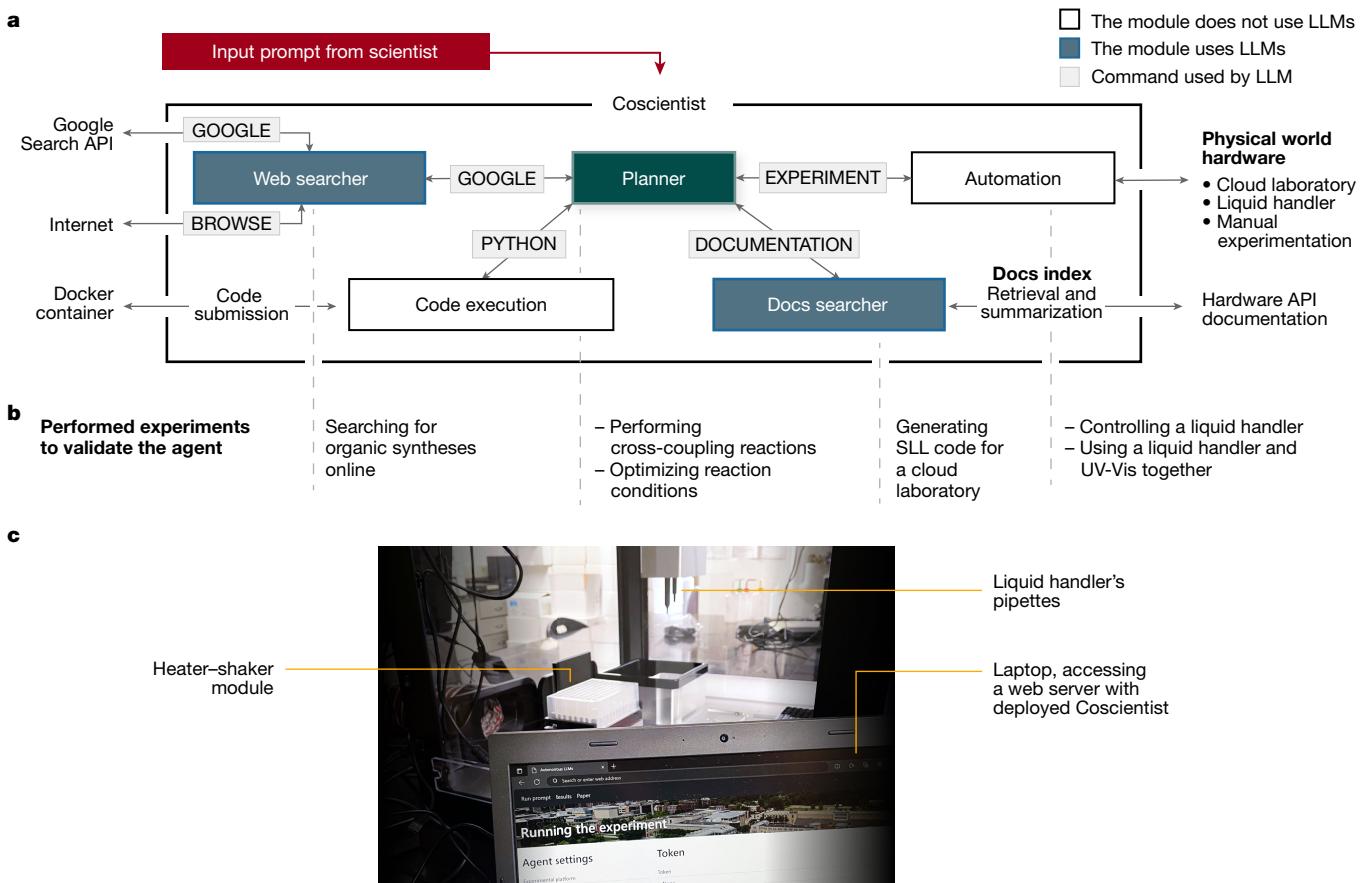


Fig. 1 | The system's architecture. **a**, Coscientist is composed of multiple modules that exchange messages. Boxes with blue background represent LLM modules, the Planner module is shown in green, and the input prompt is in red. White boxes represent modules that do not use LLMs. **b**, Types of experiments

performed to demonstrate the capabilities when using individual modules or their combinations. **c**, Image of the experimental setup with a liquid handler. UV-Vis, ultraviolet visible.

The PYTHON command allows the Planner to perform calculations to prepare the experiment using a ‘Code execution’ module. The EXPERIMENT command actualizes ‘Automation’ through APIs described by the DOCUMENTATION module. Like GOOGLE, the DOCUMENTATION command provides information to the main module from a source, in this case documentation concerning the desired API. In this study, we have demonstrated the compatibility with the OpenTrons Python API and the Emerald Cloud Lab (ECL) Symbolic Lab Language (SLL). Together, these modules make up Coscientist, which receives a simple plain text input prompt from the user (for example, “perform multiple Suzuki reactions”). This architecture is depicted in Fig. 1.

Furthermore, some of the commands can use subactions. The GOOGLE command is capable of transforming prompts into appropriate web search queries, running them against the Google Search API, browsing web pages and funneling answers back to the Planner. Similarly, the DOCUMENTATION command performs retrieval and summarization of necessary documentation (for example, robotic liquid handler or a cloud laboratory) for Planner to invoke the EXPERIMENT command.

The PYTHON command performs code execution (not reliant upon any language model) using an isolated Docker container to protect the users’ machine from any unexpected actions requested by the Planner. Importantly, the language model behind the Planner enables code to be fixed in case of software errors. The same applies to the EXPERIMENT command of the Automation module, which executes generated code on corresponding hardware or provides the synthetic procedure for manual experimentation.

Web search module

To demonstrate one of the functionalities of the Web Searcher module, we designed a test set composed of seven compounds to synthesize, as presented in Fig. 2a. The Web Searcher module versions are represented as ‘search-gpt-4’ and ‘search-gpt-3.5-turbo’. Our baselines include OpenAI’s GPT-3.5 and GPT-4, Anthropic’s Claude 1.3²⁸ and Falcon-40B-Instruct²⁹—considered one of the best open-source models at the time of this experiment as per the OpenLLM leaderboard³⁰.

We prompted every model to provide a detailed compound synthesis, ranking the outputs on the following scale (Fig. 2):

- 5 for a very detailed and chemically accurate procedure description
- 4 for a detailed and chemically accurate description but without reagent quantities
- 3 for a correct chemistry description that does not include step-by-step procedure
- 2 for extremely vague or unfeasible descriptions
- 1 for incorrect responses or failure to follow instructions
- All scores below 3 indicate task failure. It is important to note that all answers between 3 and 5 are chemically correct but offer varying levels of detail. Despite our attempts to better formalize the scale, labelling is inherently subjective and so, may be different between the labelers.

Across non-browsing models, the two versions of the GPT-4 model performed best, with Claude v.1.3 demonstrating similar performance. GPT-3 performed significantly worse, and Falcon 40B failed in most cases. All non-browsing models incorrectly synthesized ibuprofen

Article

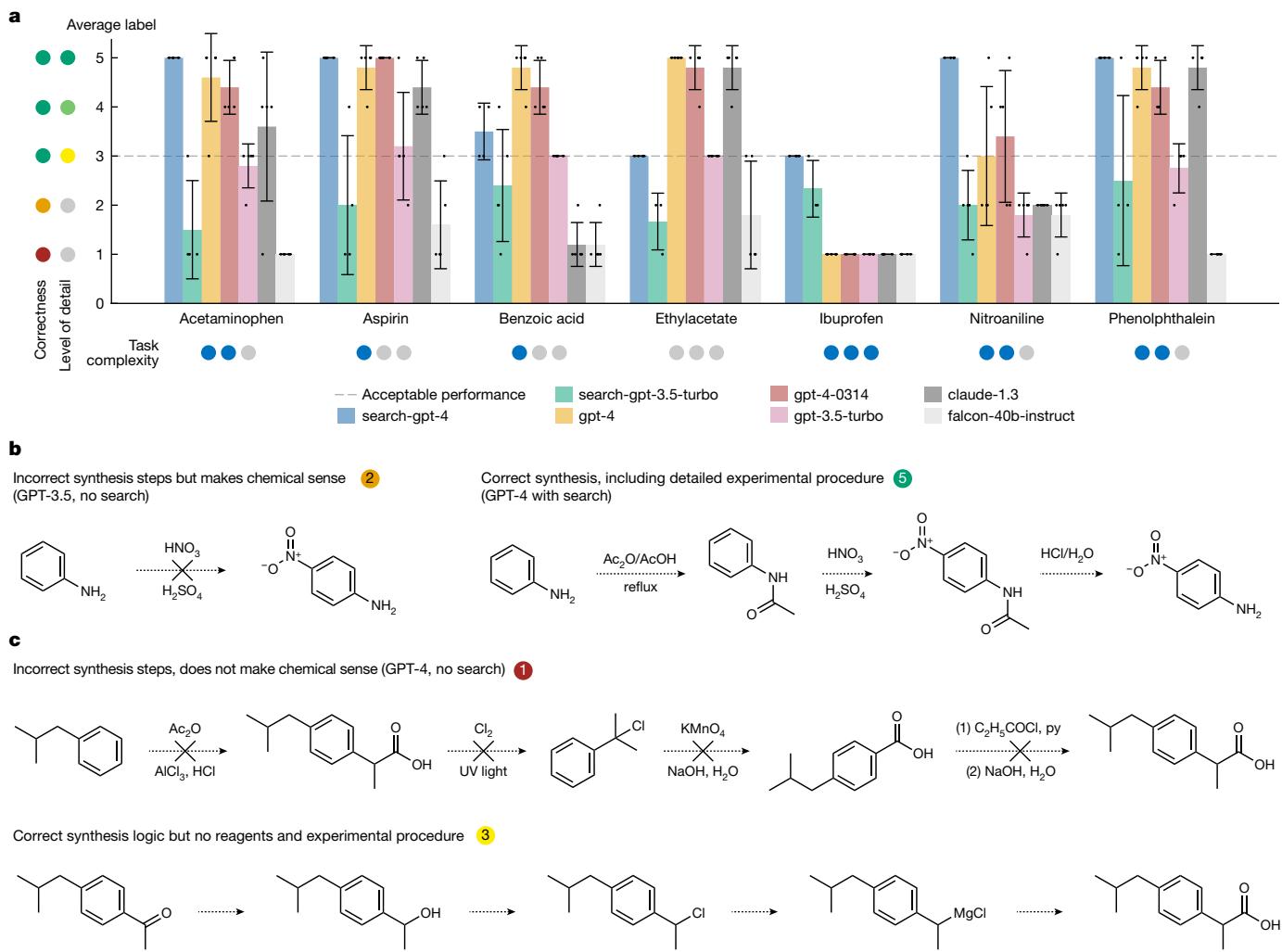


Fig. 2 | Coscientist's capabilities in chemical synthesis planning tasks. **a**, Comparison of various LLMs on compound synthesis benchmarks. Error bars represent s.d. values. **b**, Two examples of generated syntheses of nitroaniline. **c**, Two example of generated syntheses of ibuprofen. UV, ultraviolet.

(Fig. 2c). Nitroaniline is another example; although some generalization of chemical knowledge might inspire the model to propose direct nitration, this approach is not experimentally applicable as it would produce a mixture of compounds with a very minor amount of the product (Fig. 2b). Only the GPT-4 models occasionally provided the correct answer.

The GPT-4-powered Web Searcher significantly improves on synthesis planning. It reached maximum scores across all trials for acetaminophen, aspirin, nitroaniline and phenolphthalein (Fig. 2b). Although it was the only one to achieve the minimum acceptable score of three for ibuprofen, it performed lower than some of the other models for ethylacetate and benzoic acid, possibly because of the widespread nature of these compounds. These results show the importance of grounding LLMs to avoid 'hallucinations'³¹. Overall, the performance of GPT-3.5-enabled WebSearcher trailed its GPT-4 competition, mainly because of its failure to follow specific instructions regarding output format.

Extending the Planner's action space to leverage reaction databases, such as Reaxys³² or SciFinder³³, should significantly enhance the system's performance (especially for multistep syntheses). Alternatively, analysing the system's previous statements is another approach to improving its accuracy. This can be done through advanced prompting strategies, such as ReAct³⁴, Chain of Thought³⁵ and Tree of Thoughts³⁶.

Documentation search module

Addressing the complexities of software components and their interactions is crucial for integrating LLMs with laboratory automation. A key challenge lies in enabling Coscientist to effectively utilize technical documentation. LLMs can refine their understanding of common APIs, such as the Opentrons Python API³⁷, by interpreting and learning from relevant technical documentation. Furthermore, we show how GPT-4 can learn how to programme in the ECL SLL.

Our approach involved equipping Coscientist with essential documentation tailored to specific tasks (as illustrated in Fig. 3a), allowing it to refine its accuracy in using the API and improve its performance in automating experiments.

Information retrieval systems are usually based on two candidate selection approaches: inverted search index and vector database^{38–41}. For the first one, each unique word in the search index is mapped to the documents containing it. At inference time, all documents containing words from a query are selected and ranked based on various manually defined formulas⁴². The second approach starts by embedding the documents with neural networks or as term frequency-inverse document frequency embedding vectors⁴³, followed by the construction of a vector database. Retrieval of similar vectors from this database occurs at inference time, usually using one of the approximate nearest neighbour search algorithms⁴⁴. When strategies such as Transformer

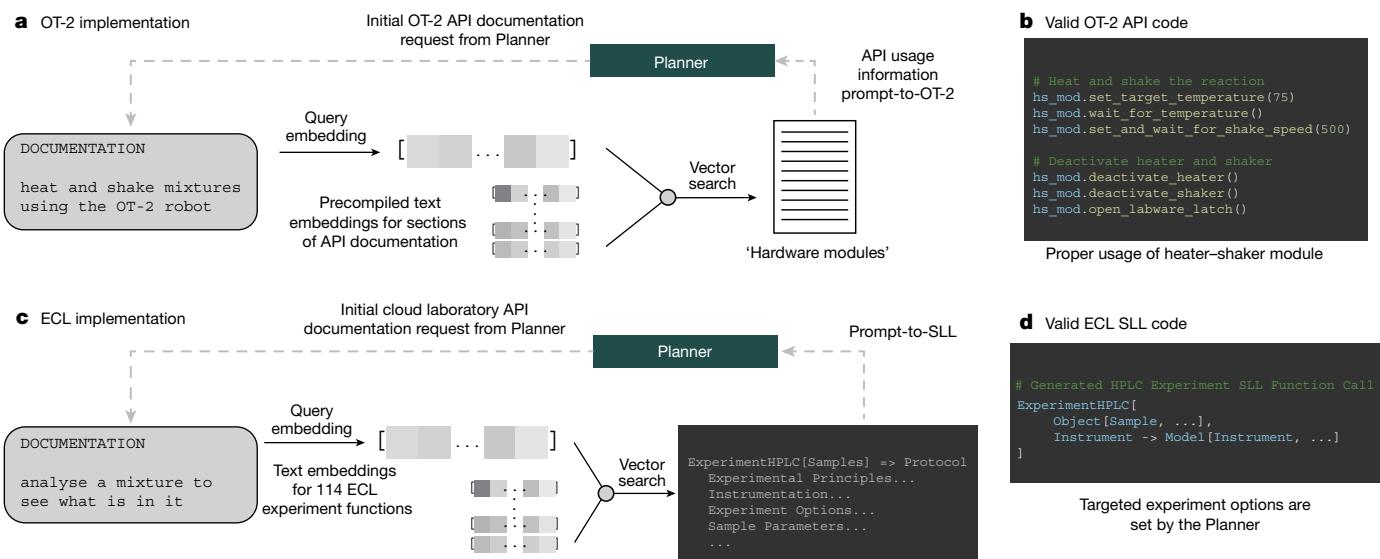


Fig. 3 | Overview of documentation search. **a**, Prompt-to-code through ada embedding and distance-based vector search. **b**, Example of code for using OT-2's heater–shaker module. **c**, Prompt-to-function/prompt-to-SLL (to symbolic

laboratory language) through supplementation of documentation. **d**, Example of valid ECL SLL code for performing high-performance liquid chromatography (HPLC) experiments.

models are used, there are more chances to account for synonyms natively without doing synonym-based query expansion, as would be done in the first approach⁴⁵.

Following the second approach, all sections of the OT-2 API documentation were embedded using OpenAI's ada model. To ensure proper use of the API, an ada embedding for the Planner's query was generated, and documentation sections are selected through a distance-based vector search. This approach proved critical for providing Coscientist with information about the heater–shaker hardware module necessary for performing chemical reactions (Fig. 3b).

A greater challenge emerges when applying this approach to a more diverse robotic ecosystem, such as the ECL. Nonetheless, we can explore the effectiveness of providing information about the ECL SLL, which is currently unknown to the GPT-4 model. We conducted three separate investigations concerning the SLL: (1) prompt-to-function; (2) prompt-to-SLL; and (3) prompt-to-samples. Those investigations are detailed in Supplementary Information section 'ECL experiments'.

For investigation 1, we provide the Docs searcher with a documentation guide from ECL pertaining to all available functions for running experiments⁴⁶. Figure 3c summarizes an example of the user providing a simple prompt to the system, with the Planner receiving relevant ECL functions. In all cases, functions are correctly identified for the task.

Figure 3c,d continues to describe investigation 2, the prompt-to-SLL investigation. A single appropriate function is selected for the task, and the documentation is passed through a separate GPT-4 model to perform code retention and summarization. After the complete documentation has been processed, the Planner receives usage information to provide EXPERIMENT code in the SLL. For instance, we provide a simple example that requires the 'ExperimentHPLC' function. Proper use of this function requires familiarity with specific 'Models' and 'Objects' as they are defined in the SLL. Generated code was successfully executed at ECL; this is available in Supplementary Information. The sample was a caffeine standard sample. Other parameters (column, mobile phases, gradients) were determined by ECL's internal software (a high-level description is in Supplementary Information section 'HPLC experiment parameter estimation'). Results of the experiment are provided in Supplementary Information section 'Results of the HPLC experiment in the cloud lab'. One can see that the air bubble

was injected along with the analyte's solution. This demonstrates the importance of development of automated techniques for quality control in cloud laboratories. Follow-up experiments leveraging web search to specify and/or refine additional experimental parameters (column chemistry, buffer system, gradient and so on) would be required to optimize the experimental results. Further details on this investigation are in Supplementary Information section 'Analysis of ECL documentation search results'.

A separate prompt-to-samples investigation, investigation 3, was conducted by providing a catalogue of available samples, enabling the identification of relevant stock solutions that are on ECL's shelves. To showcase this feature, we provide the Docs searcher module with all 1,110 Model samples from the catalogue. By simply providing a search term (for example, 'Acetonitrile'), all relevant samples are returned. This is also available in Supplementary Information.

Controlling laboratory hardware

Access to documentation enables us to provide sufficient information for Coscientist to conduct experiments in the physical world. To initiate the investigation, we chose the Opentrons OT-2, an open-source liquid handler with a well-documented Python API. The 'Getting Started' page from its documentation was supplied to the Planner in the system prompt. Other pages were vectorized using the approach described above. For this investigation, we did not grant access to the internet (Fig. 4a).

We started with simple plate layout-specific experiments. Straightforward prompts in natural language, such as "colour every other line with one colour of your choice", resulted in accurate protocols. When executed by the robot, these protocols closely resembled the requested prompt (Fig. 4b–e).

Ultimately, we aimed to assess the system's ability to integrate multiple modules simultaneously. Specifically, we provided the 'UVVIS' command, which can be used to pass a microplate to plate reader working in the ultraviolet–visible wavelength range. To evaluate Coscientist's capabilities to use multiple hardware tools, we designed a toy task; in 3 wells of a 96-well plate, three different colours are present—red, yellow and blue. The system must determine the colours and their positions on the plate without any prior information.

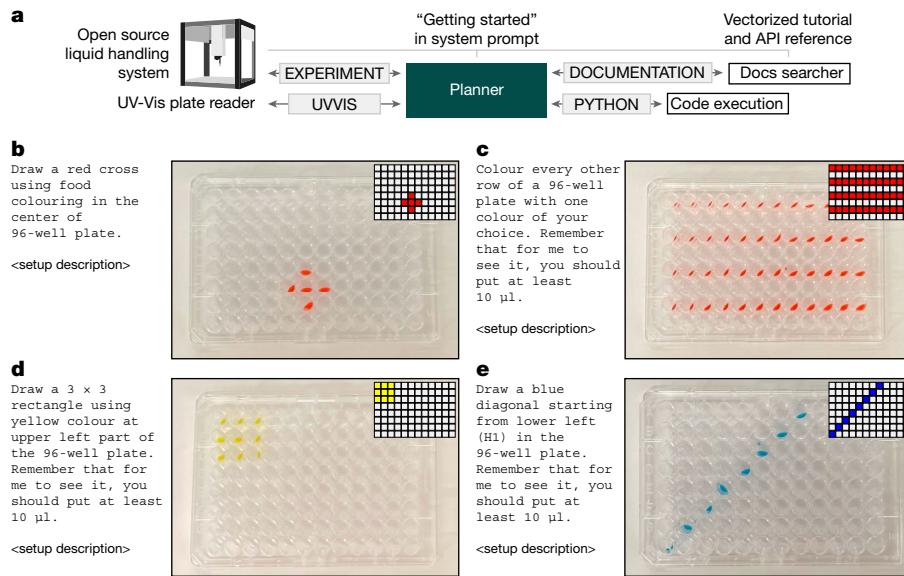


Fig. 4 | Robotic liquid handler control capabilities and integration with analytical tools. a, Overview of Coscientist's configuration. **b,** Drawing a red cross. **c,** Colouring every other row. **d,** Drawing a yellow rectangle. **e,** Drawing a blue diagonal.

The Coscientist's first action was to prepare small samples of the original solutions (Extended Data Fig. 1). Ultraviolet-visible measurements were then requested to be performed by the Coscientist (Supplementary Information section 'Solving the colours problem' and Supplementary Fig. 1). Once completed, Coscientist was provided with a file name containing a NumPy array with spectra for each well of the microplate. Coscientist subsequently generated Python code to identify the wavelengths with maximum absorbance and used these data to correctly solve the problem, although it required a guiding prompt asking it to think through how different colours absorb light.

Integrated chemical experiment design

We evaluated Coscientist's ability to plan catalytic cross-coupling experiments by using data from the internet, performing the necessary calculations and ultimately, writing code for the liquid handler. To increase complexity, we asked Coscientist to use the OT-2 heater–shaker module released after the GPT-4 training data collection cutoff. The available commands and actions supplied to the Coscientist are shown in Fig. 5a. Although our setup is not yet fully automated (plates were moved manually), no human decision-making was involved.

The test challenge for Coscientist's complex chemical experimentation capabilities was designed as follows. (1) Coscientist is provided with a liquid handler equipped with two microplates (source and target plates). (2) The source plate contains stock solutions of multiple reagents, including phenyl acetylene and phenylboronic acid, multiple aryl halide coupling partners, two catalysts, two bases and the solvent to dissolve the sample (Fig. 5b). (3) The target plate is installed on the OT-2 heater–shaker module (Fig. 5c). (4) Coscientist's goal is to successfully design and perform a protocol for Suzuki–Miyaura and Sonogashira coupling reactions given the available resources.

To start, Coscientist searches the internet for information on the requested reactions, their stoichiometries and conditions (Fig. 5d). The correct coupling partners are selected for the corresponding reactions. Designing and performing the requested experiments, the strategy of Coscientist changes among runs (Fig. 5f). Importantly, the system does not make chemistry mistakes (for instance, it never selects phenylboronic acid for the Sonogashira reaction). Interestingly, the

base DBU (1,8-diazabicyclo[5.4.0]undec-7-ene) is selected more often with the PEPPSI–IPr (PEPPSI, pyridine-enhanced precatalyst preparation stabilization and initiation; IPr, 1,3-bis(2,6-diisopropylphenyl)imidazol-2-ylidene) complex, with that preference switching in Sonogashira reaction experiments; likewise, bromobenzene is chosen more often for Suzuki than for Sonogashira couplings. Additionally, the model can provide justifications on specific choices (Fig. 5g), demonstrating the ability to operate with concepts such as reactivity and selectivity (more details are in Supplementary Information section 'Analysis of behaviour across multiple runs'). This capability highlights a potential future use case to analyse the reasoning of the LLMs used by performing experiments multiple times. Although the Web Searcher visited various websites (Fig. 5h), overall Coscientist retrieves Wikipedia pages in approximately half of cases; notably, American Chemical Society and Royal Society of Chemistry journals are amongst the top five sources.

Coscientist then calculates the required volumes of all reactants and writes a Python protocol for running the experiment on the OT-2 robot. However, an incorrect heater–shaker module method name was used. Upon making this mistake, Coscientist uses the Docs searcher module to consult the OT-2 documentation. Next, Coscientist modifies the protocol to a corrected version, which ran successfully (Extended Data Fig. 2). Subsequent gas chromatography–mass spectrometry analysis of the reaction mixtures revealed the formation of the target products for both reactions. For the Suzuki reaction, there is a signal in the chromatogram at 9.53 min where the mass spectra match the mass spectra for biphenyl (corresponding molecular ion mass-to-charge ratio and fragment at 76 Da) (Fig. 5i). For the Sonogashira reaction, we see a signal at 12.92 min with a matching molecular ion mass-to-charge ratio; the fragmentation pattern also looks very close to the one from the spectra of the reference compound (Fig. 5j). Details are in Supplementary Information section 'Results of the experimental study'.

Although this example requires Coscientist to reason on which reagents are most suitable, our experimental capabilities at that point limited the possible compound space to be explored. To address this, we performed several computational experiments to evaluate how a similar approach can be used to retrieve compounds from large compound libraries⁴⁷. Figure 5e shows Coscientist's performance across five common organic transformations, with outcomes depending on the

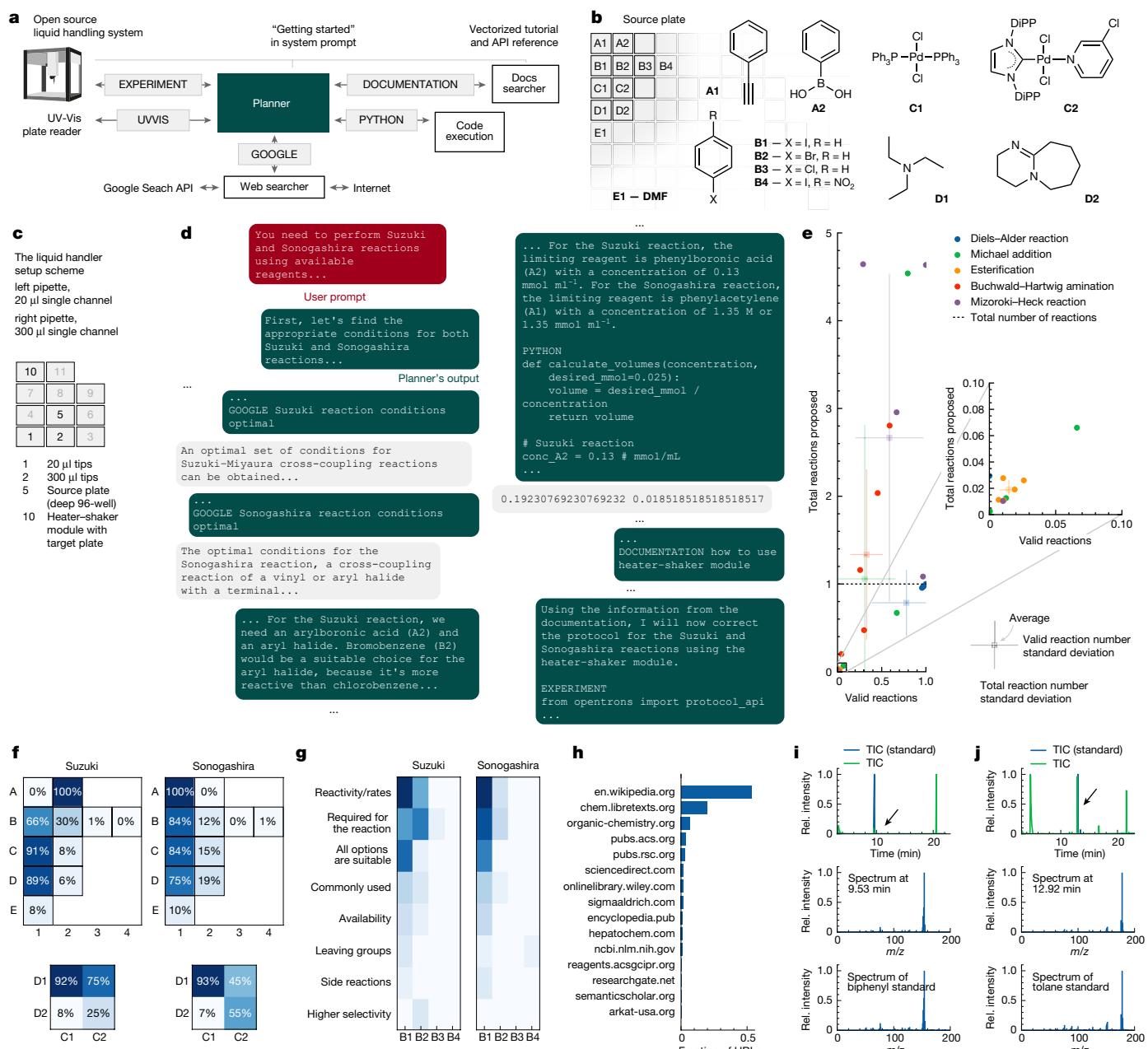


Fig. 5 | Cross-coupling Suzuki and Sonogashira reaction experiments designed and performed by Coscientist. **a**, Overview of Coscientist's configuration. **b**, Available compounds (DMF, dimethylformamide; DiPP, 2,6-diisopropylphenyl). **c**, Liquid handler setup. **d**, Solving the synthesis problem. **e**, Comparison of reagent selection performance with a large dataset. **f**, Comparison of reagent choices across multiple runs. **g**, Overview of justifications made when selecting various aryl halides. **h**, Frequency of

queried reaction and its specific run (the GitHub repository has more details). For each reaction, Coscientist was tasked with generating reactions for compounds from a simplified molecular-input line-entry system (SMILES) database. To achieve the task, Coscientist uses web search and code execution with the RDKit chemoinformatics package.

Chemical reasoning capabilities

The system demonstrates appreciable reasoning capabilities, enabling the request of necessary information, solving of multistep problems and generation of code for experimental design. Some researchers

believe that the community is only starting to understand all the capabilities of GPT-4 (ref. 48). OpenAI has shown that GPT-4 could rely on some of those capabilities to take actions in the physical world during their initial red team testing performed by the Alignment Research Center¹⁴.

One of the possible strategies to evaluate an intelligent agent's reasoning capabilities is to test if it can use previously collected data to guide future actions. Here, we focused on the multi-variable design and optimization of Pd-catalysed transformations, showcasing Coscientist's abilities to tackle real-world experimental campaigns involving thousands of examples. Instead of connecting LLMs to an

Article

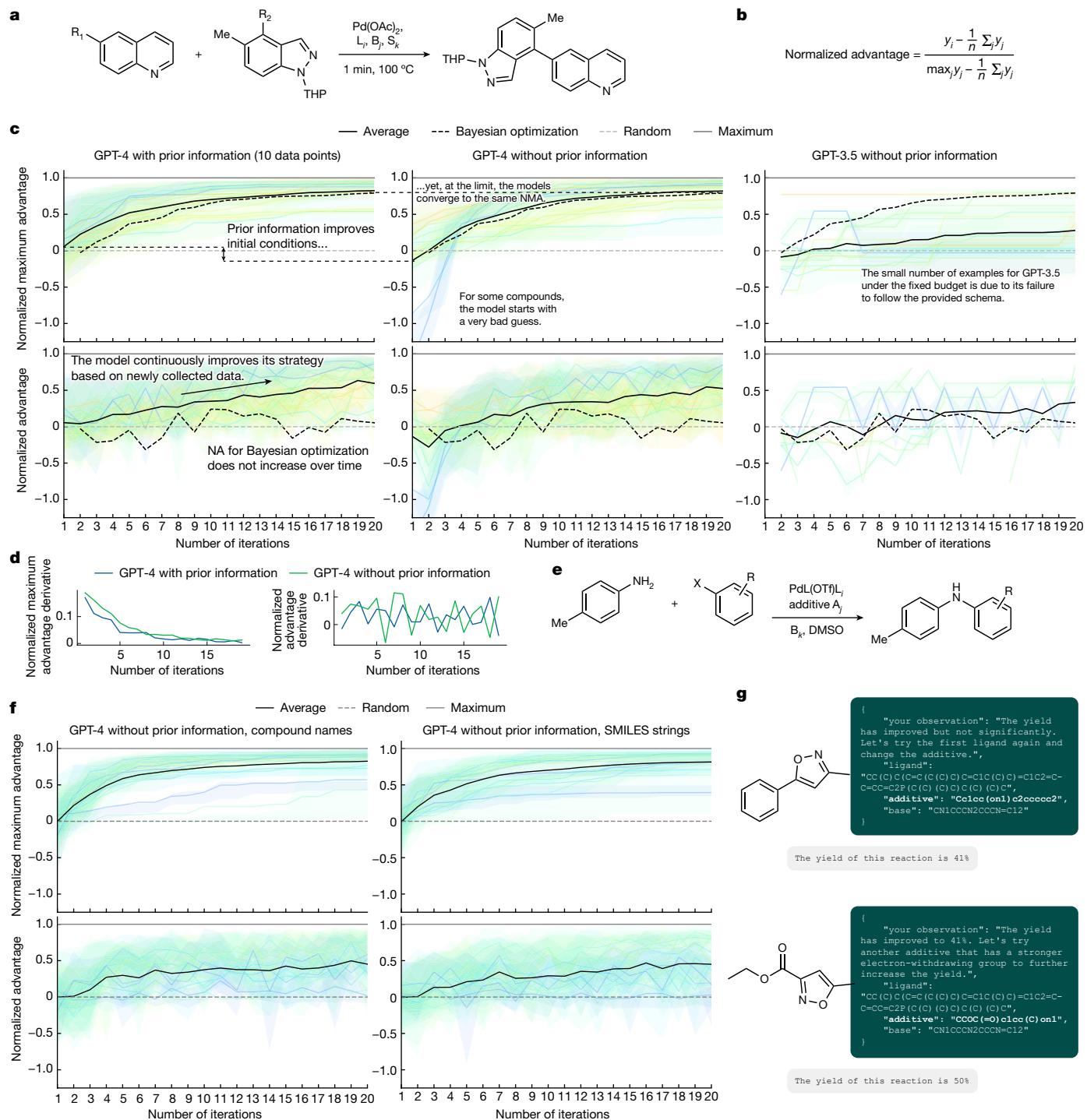


Fig. 6 | Results of the optimization experiments. **a**, A general reaction scheme from the flow synthesis dataset analysed in **c** and **d**. **b**, The mathematical expression used to calculate normalized advantage values. **c**, Comparison of the three approaches (GPT-4 with prior information, GPT-4 without prior information and GPT-3.5 without prior information) used to perform the optimization process. **d**, Derivatives of the NMA and normalized advantage

values evaluated in **c**, left and centre panels. **e**, Reaction from the C–N cross-coupling dataset analysed in **f** and **g**. **f**, Comparison of two approaches using compound names and SMILES string as compound representations. **g**, Coscientist can reason about electronic properties of the compounds, even when those are represented as SMILES strings. DMSO, dimethyl sulfoxide.

optimization algorithm as previously done by Ramos et al.⁴⁹, we aimed to use Coscientist directly.

We selected two datasets containing fully mapped reaction condition spaces where yield was available for all combinations of variables. One is a Suzuki reaction dataset collected by Perera et al.⁵⁰, where these reactions were performed in flow with varying ligands, reagents/bases and solvents (Fig. 6a). Another is Doyle's Buchwald–Hartwig reaction

dataset⁵¹ (Fig. 6e), where variations in ligands, additives and bases were recorded. At this point, any reaction proposed by Coscientist would be within these datasets and accessible as a lookup table.

We designed the Coscientist's chemical reasoning capabilities test as a game with the goal of maximizing the reaction yield. The game's actions consisted of selecting specific reaction conditions with a sensible chemical explanation while listing the player's observations

about the outcome of the previous iteration. The only hard rule was for the player to provide its actions written in JavaScript Object Notation (JSON) format. If the JSON file could not be parsed, the player is alerted of its failure to follow the specified data format. The player had a maximum of 20 iterations (accounting for 5.2% and 6.9% of the total space for the first and second datasets, respectively) to finish the game.

We evaluate Coscientist's performance using the normalized advantage metric (Fig. 6b). Advantage is defined as the difference between a given iteration yield and the average yield (advantage over a random strategy). Normalized advantage measures the ratio between advantage and maximum advantage (that is, the difference between the maximum and average yield). The normalized advantage metric has a value of one if the maximum yield is reached, zero if the system exhibits completely random behaviour and less than zero if the performance at this step is worse than random. An increase in normalized advantage over each iteration demonstrates Coscientist's chemical reasoning capabilities. The best result for a given iteration can be evaluated using the normalized maximum advantage (NMA), which is the normalized value of the maximum advantage achieved until the current step. As NMA cannot decrease, the valuable observations come in the form of the rate of its increase and its final point. Finally, during the first step, the values for NMA and normalized advantage equal each other, portraying the model's prior knowledge (or lack thereof) without any data being collected.

For the Suzuki dataset, we compared three separate approaches: (1) GPT-4 with prior information included in the prompt (which consisted of 10 yields from random combinations of reagents); (2) GPT-4; or (3) GPT-3.5 without any prior information (Fig. 6c). When comparing GPT-4 with the inclusion and exclusion of prior information, it is clear that the initial guess for the former scenario is better, which aligns with our expectations considering the provided information about the system's reactivity. Notably, when excluding prior information, there are some poor initial guesses, whereas there are none when the model has prior information. However, at the limit, the models converge to the same NMA. The GPT-3.5 model plots have a very limited number of data points, primarily because of its inability to output messages in the correct JSON schema as requested in the prompt. It is unclear if the GPT-4 training data contain any information from these datasets. If so, one would expect that the initial model guess would be better than what we observed.

The normalized advantage values increase over time, suggesting that the model can effectively reuse the information obtained to provide more specific guidance on reactivity. Evaluating the derivative plots (Fig. 6d) does not show any significant difference between instances with and without the input of prior information.

There are many established optimization algorithms for chemical reactions. In comparison with standard Bayesian optimization⁵², both GPT-4-based approaches show higher NMA and normalized advantage values (Fig. 6c). A detailed overview of the exact Bayesian optimization strategy used is provided in Supplementary Information section 'Bayesian optimization procedure'. It is observed that Bayesian optimization's normalized advantage line stays around zero and does not increase over time. This may be caused by different exploration/exploitation balance for these two approaches and may not be indicative of their performance. For this purpose, the NMA plot should be used. Changing the number of initial samples does not improve the Bayesian optimization trajectory (Extended Data Fig. 3a). Finally, this performance trend is observed for each unique substrate pairings (Extended Data Fig. 3b).

For the Buchwald–Hartwig dataset (Fig. 6e), we compared a version of GPT-4 without prior information operating over compound names or over compound SMILES strings. It is evident that both instances have very similar performance levels (Fig. 6f). However, in certain scenarios, the model demonstrates the ability to reason about the reactivity of these compounds simply by being provided their SMILES strings (Fig. 6g).

Discussion

In this paper, we presented a proof of concept for an artificial intelligent agent system capable of (semi-)autonomously designing, planning and multistep executing scientific experiments. Our system demonstrates advanced reasoning and experimental design capabilities, addressing complex scientific problems and generating high-quality code. These capabilities emerge when LLMs gain access to relevant research tools, such as internet and documentation search, coding environments and robotic experimentation platforms. The development of more integrated scientific tools for LLMs has potential to greatly accelerate new discoveries.

The development of new intelligent agent systems and automated methods for conducting scientific experiments raises potential concerns about the safety and potential dual-use consequences, particularly in relation to the proliferation of illicit activities and security threats. By ensuring the ethical and responsible use of these powerful tools, we are continuing to explore the vast potential of LLMs in advancing scientific research while mitigating the risks associated with their misuse. A brief dual-use study of Coscientist is provided in Supplementary Information section 'Safety implications: Dual-use study'.

Technology use disclosure

The writing of the preprint version of this manuscript was assisted by ChatGPT (specifically, GPT-4 being used for grammar and typos). All authors have read, corrected and verified all information presented in this manuscript and Supplementary Information.

Online content

Any methods, additional references, Nature Portfolio reporting summaries, source data, extended data, supplementary information, acknowledgements, peer review information; details of author contributions and competing interests; and statements of data and code availability are available at <https://doi.org/10.1038/s41586-023-06792-0>.

1. Brown, T. et al. in *Advances in Neural Information Processing Systems* Vol. 33 (eds Larochelle, H. et al.) 1877–1901 (Curran Associates, 2020).
2. Thoppilan, R. et al. LaMDA: language models for dialog applications. Preprint at <https://arxiv.org/abs/2201.08239> (2022).
3. Touvron, H. et al. LLaMA: open and efficient foundation language models. Preprint at <https://arxiv.org/abs/2302.13971> (2023).
4. Hoffmann, J. et al. Training compute-optimal large language models. In *Advances in Neural Information Processing Systems* 30016–30030 (NeurIPS, 2022).
5. Chowdhery, A. et al. PaLM: scaling language modeling with pathways. *J. Mach. Learn. Res.* **24**, 1–113 (2022).
6. Lin, Z. et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science* **379**, 1123–1130 (2023).
7. Luo, R. et al. BioGPT: generative pre-trained transformer for biomedical text generation and mining. *Brief Bioinform.* **23**, bbac409 (2022).
8. Irwin, R., Dimitriadis, S., He, J. & Bjerrum, E. J. Chemformer: a pre-trained transformer for computational chemistry. *Mach. Learn. Sci. Technol.* **3**, 015022 (2022).
9. Kim, H., Na, J. & Lee, W. B. Generative chemical transformer: neural machine learning of molecular geometric structures from chemical language via attention. *J. Chem. Inf. Model.* **61**, 5804–5814 (2021).
10. Jablonka, K. M., Schwaller, P., Ortega-Guerrero, A. & Smit, B. Leveraging large language models for predictive chemistry. Preprint at <https://chemrxiv.org/engage/chemrxiv/article-details/652e50b98bab5d2055852dde> (2023).
11. Xu, F. F., Alon, U., Neubig, G. & Hellendoorn, V. J. A systematic evaluation of large language models of code. In *Proc. 6th ACM SIGPLAN International Symposium on Machine Programming* 1–10 (ACM, 2022).
12. Nijkamp, E. et al. CodeGen: an open large language model for code with multi-turn program synthesis. In *Proc. 11th International Conference on Learning Representations* (ICLR, 2022).
13. Kaplan, J. et al. Scaling laws for neural language models. Preprint at <https://arxiv.org/abs/2001.08361> (2020).
14. OpenAI. *GPT-4 Technical Report* (OpenAI, 2023).
15. Ziegler, D. M. et al. Fine-tuning language models from human preferences. Preprint at <https://arxiv.org/abs/1909.08593> (2019).
16. Ouyang, L. et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems* 27730–27744 (NeurIPS, 2022).

17. Granda, J. M., Donina, L., Dragone, V., Long, D.-L. & Cronin, L. Controlling an organic synthesis robot with machine learning to search for new reactivity. *Nature* **559**, 377–381 (2018).
18. Caramelli, D. et al. Discovering new chemistry with an autonomous robotic platform driven by a reactivity-seeking neural network. *ACS Cent. Sci.* **7**, 1821–1830 (2021).
19. Angello, N. H. et al. Closed-loop optimization of general reaction conditions for heteroaryl Suzuki–Miyaura coupling. *Science* **378**, 399–405 (2022).
20. Adamo, A. et al. On-demand continuous-flow production of pharmaceuticals in a compact, reconfigurable system. *Science* **352**, 61–67 (2016).
21. Coley, C. W. et al. A robotic platform for flow synthesis of organic compounds informed by AI planning. *Science* **365**, eaax1566 (2019).
22. Burger, B. et al. A mobile robotic chemist. *Nature* **583**, 237–241 (2020).
23. Auto-GPT: the heart of the open-source agent ecosystem. *GitHub* <https://github.com/Sigificant-Gravitas/AutoGPT> (2023).
24. BabyAGI. *GitHub* <https://github.com/yoheinakajima/babyagi> (2023).
25. Chase, H. LangChain. *GitHub* <https://github.com/langchain-ai/langchain> (2023).
26. Bran, A. M., Cox, S., White, A. D. & Schwaller, P. ChemCrow: augmenting large-language models with chemistry tools. Preprint at <https://arxiv.org/abs/2304.05376> (2023).
27. Liu, P. et al. Pre-train, prompt, and predict: a systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.* **55**, 195 (2021).
28. Bai, Y. et al. Constitutional AI: harmlessness from AI feedback. Preprint at <https://arxiv.org/abs/2212.08073> (2022).
29. Falcon LLM. *TII* <https://falconllm.tii.ai> (2023).
30. Open LLM Leaderboard. *Hugging Face* https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard (2023).
31. Ji, Z. et al. Survey of hallucination in natural language generation. *ACM Comput. Surv.* **55**, 248 (2023).
32. Reaxys <https://www.reaxys.com> (2023).
33. SciFinder <https://scifinder.cas.org> (2023).
34. Yao, S. et al. ReAct: synergizing reasoning and acting in language models. In *Proc. 11th International Conference on Learning Representations* (ICLR, 2022).
35. Wei, J. et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems* 24824–24837 (NeurIPS, 2022).
36. Long, J. Large language model guided tree-of-thought. Preprint at <https://arxiv.org/abs/2305.08291> (2023).
37. Opentrons Python Protocol API. *Opentrons* <https://docs.opentrons.com/v2/> (2023).
38. Tu, Z. et al. Approximate nearest neighbor search and lightweight dense vector reranking in multi-stage retrieval architectures. In *Proc. 2020 ACM SIGIR on International Conference on Theory of Information Retrieval* 97–100 (ACM, 2020).
39. Lin, J. et al. Pyserini: a python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proc. 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* 2356–2362 (ACM, 2021).
40. Qadrud-Din, J. et al. Transformer based language models for similar text retrieval and ranking. Preprint at <https://arxiv.org/abs/2005.04588> (2020).
41. Paper QA. *GitHub* <https://github.com/whitehead/paper-qa> (2023).
42. Robertson, S. & Zaragoza, H. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retrieval* **3**, 333–389 (2009).
43. Data Mining. *Mining of Massive Datasets* (Cambridge Univ., 2011).
44. Johnson, J., Douze, M. & Jegou, H. Billion-scale similarity search with GPUs. *IEEE Trans. Big Data* **7**, 535–547 (2021).
45. Vechtomova, O. & Wang, Y. A study of the effect of term proximity on query expansion. *J. Inf. Sci.* **32**, 324–333 (2006).
46. Running experiments. *Emerald Cloud Lab* <https://www.emeraldcloudlab.com/guides/runningexperiments> (2023).
47. Sanchez-Garcia, R. et al. CoPriNet: graph neural networks provide accurate and rapid compound price prediction for molecule prioritisation. *Digital Discov.* **2**, 103–111 (2023).
48. Bubeck, S. et al. Sparks of artificial general intelligence: early experiments with GPT-4. Preprint at <https://arxiv.org/abs/2303.12712> (2023).
49. Ramos, M. C., Michavly, S. S., Porosoff, M. D. & White, A. D. Bayesian optimization of catalysts with in-context learning. Preprint at <https://arxiv.org/abs/2304.05341> (2023).
50. Perera, D. et al. A platform for automated nanomole-scale reaction screening and micromole-scale synthesis in flow. *Science* **359**, 429–434 (2018).
51. Ahneman, D. T., Estrada, J. G., Lin, S., Dreher, S. D. & Doyle, A. G. Predicting reaction performance in C–N cross-coupling using machine learning. *Science* **360**, 186–190 (2018).
52. Hickman, R. et al. Atlas: a brain for self-driving laboratories. Preprint at <https://chemrxiv.org/engage/chemrxiv/article-details/64f6560579853bbd781bcef6> (2023).

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

© The Author(s) 2023

Data availability

Examples of the experiments discussed in the text are provided in the Supplementary Information. Because of safety concerns, data, code and prompts will be only fully released after the development of US regulations in the field of artificial intelligence and its scientific applications. Nevertheless, the outcomes of this work can be reproduced using actively developed frameworks for autonomous agent development. The reviewers had access to the web application and were able to verify any statements related to this work. Moreover, we provide a simpler implementation of the described approach, which, although it may not produce the same results, allows for deeper understanding of the strategies used in this work.

Code availability

Simpler implementation as well as generated outputs used for quantitative analysis are provided at <https://github.com/gomesgroup/coscientist>.

Acknowledgements We thank the following Carnegie Mellon University Chemistry groups for their assistance with providing the chemicals needed for the Coscientist's experiments: Sydlik, Garcia Borsch, Matyjaszewski and Ly. We give special thanks to the Noonan group (K. Noonan and D. Sharma) for providing access to chemicals and gas chromatography-mass spectrometry analysis. We also thank the team at Emerald Cloud Lab (with special attention

to Y. Benslimane, H. Gronlund, B. Smith and B. Frezza) for assisting us with parsing their documentation and executing experiments. G.G. is grateful to the Carnegie Mellon University Cloud Lab Initiative led by the Mellon College of Science for its vision of the future of physical sciences. G.G. thanks Carnegie Mellon University, the Mellon College of Sciences and its Department of Chemistry; and the College of Engineering and its Department of Chemical Engineering for the start-up support. D.A.B. was partially funded by the National Science Foundation Center for Chemoenzymatic Synthesis (Grant no. 2221346). R.M. was funded by the National Science Foundation Center for Computer-Assisted Synthesis (Grant no. 2202693).

Author contributions D.A.B. designed the computational pipeline and developed the 'Planner', 'Web searcher' and 'Code execution' modules. R.M. assisted in designing the computational pipeline and developed the 'Docs searcher' module. B.K. analysed the behaviours of the Docs searcher module to enable Coscientist to produce experiment code in Emerald Cloud Lab's Symbolic Lab Language. D.A.B. assisted and oversaw Coscientist's chemistry experiments. D.A.B., R.M. and G.G. designed and performed initial computational safety studies. D.A.B. designed and graded Coscientist's synthesis capabilities study. D.A.B. co-designed with G.G. and performed the optimization experiments. R.M. performed the large compound library experiment and Bayesian optimization baseline runs. G.G. designed the concepts, performed preliminary studies and supervised the project. D.A.B., R.M. and G.G. wrote this manuscript.

Competing interests G.G. is part of the AI Scientific Advisory Board of Emerald Cloud Lab. Experiments and conclusions in this manuscript were made before G.G.'s appointment to this role. B.K. is an employee of Emerald Cloud Lab. D.A.B. and G.G. are co-founders of aithera.ai, a company focusing on responsible use of artificial intelligence for research.

Additional information

Supplementary information The online version contains supplementary material available at <https://doi.org/10.1038/s41586-023-06792-0>.

Correspondence and requests for materials should be addressed to Gabe Gomes.

Peer review information *Nature* thanks Sebastian Farquhar, Tiago Rodrigues and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.

Reprints and permissions information is available at <http://www.nature.com/reprints>.

Article

You have a source plate with three samples in unknown positions (yellow, blue, and red), each potentially located in one of the wells (A1, B1, and C1). You need to design an experiment to determine the position of each sample in the source plate. <setup description>

User prompt

Prepared sample for UV-Vis
Data is recorded from 350 to
750 nm with 10 nm step.



First, I will transfer a small amount of each sample from the source plate to separate wells in the target plate. Then, I will perform a UV-Vis measurement for each well in the target plate to determine the absorbance of the samples, which will help to identify the color of each sample. Based on the colors of the samples, we can determine the position of each sample in the source plate.

EXPERIMENT
from opentrons import protocol_api
...

Perform UV-Vis measurement on the target plate
UVVIS plate 1

In the code the Agent moves liquids from A1, B1, and C1 to A1, A2, and A3. Then it dilutes them.

Planner's output

UV-Vis data from plate 1 is ready. You can access it using PYTHON, it will be in the file named 'plate1.npy'. You will have NumPy installed. The file contains NumPy array of shape (8, 12, 41). There are 41 wavelengths from 350 to 750 with 10 nm step. Think about how would different colors absorb first.

Automated answer
+ guiding prompt

Different colors absorb light at different wavelengths. Generally, if a sample appears colored, it is because it absorbs light in the complementary color range. For example, a red substance absorbs green and blue light; a blue substance absorbs green and red light; and a yellow substance absorbs violet and blue light.

First, I will load the UV-Vis data from the file 'plate1.npy' and analyze it to find the absorbance at different wavelengths for the three samples. Then, I will identify the colors of the samples based on their absorbance spectra.

PYTHON
import numpy as np
...
print(samples_max_wavelength_nm)

[530, 430, 630]

Automated answer

The maximum absorbance wavelengths for the three samples are 530 nm, 430 nm, and 630 nm. Based on the absorbance spectra, we can deduce their colors:

1. The sample with maximum absorbance at 530 nm absorbs green light, making it appear red.
2. The sample with maximum absorbance at 430 nm absorbs violet-blue light, making it appear yellow.
3. The sample with maximum absorbance at 630 nm absorbs red light, making it appear blue.

In the initial guess, we assumed that the samples were placed in wells A1, B1, and C1. However, based on the absorbance data analysis, we find:

1. The red sample is located in well A1.
2. The yellow sample is located in well B1.
3. The blue sample is located in well C1.

Extended Data Fig. 1 | Using UV-Vis and liquid handler to solve food colouring identification problem. Guiding prompt in the third message is shown in bold. In the first message the user prompt is provided, then code for sample

preparation is generated, resulting data is provided as NumPy array, which is then analysed to give the final answer.

```

1 from opentrons import protocol_api
2
3 metadata = {
4     'apiLevel': '2.13'
5 }
6
7 def run(protocol: protocol_api.ProtocolContext):
8     # Load labware
9     hs_mod = protocol.load_module('heaterShakerModuleV1', 10)
10    reagents = protocol.load_labware('nest_96_wellplate_2ml_deep', 5)
11    reaction_plate = hs_mod.load_labware('nest_96_wellplate_2ml_deep')
12    tiprack20 = protocol.load_labware('opentrons_96_tiprack_20ul', 1)
13    tiprack300 = protocol.load_labware('opentrons_96_tiprack_300ul', 2)
14
15    # Load pipettes
16    p20 = protocol.load_instrument('p20_single_gen2', 'left', tip_racks=[tiprack20])
17    p300 = protocol.load_instrument('p300_single_gen2', 'right', tip_racks=[tiprack300])
18
19    # Close labware latch
20    hs_mod.close_labware_latch()
21
22    # Transfer reagents for Suzuki reaction (0.025 mmol scale)
23
24    ## Transfer reagents for Sonogashira reaction (0.025 mmol scale)
25    p20.pick_up_tip()
26    p20.aspirate(18.5, reagents['A1'])
27    p20.dispense(18.5, reaction_plate['B1'])
28    p20.drop_tip()
29
30    p20.pick_up_tip()
31    p20.aspirate(13.4, reagents['B1'])
32    p20.dispense(13.4, reaction_plate['B1'])
33    p20.drop_tip()
34
35    p300.pick_up_tip()
36    p300.aspirate(83.3, reagents['C2'])
37    p300.dispense(83.3, reaction_plate['B1'])
38    p300.drop_tip()
39
40    p20.pick_up_tip()
41    p20.aspirate(17.7, reagents['D1'])
42    p20.dispense(17.7, reaction_plate['B1'])
43    p20.drop_tip()
44
45    # Heat and shake the reactions
46    hs_mod.set_target_temperature(75)
47    hs_mod.wait_for_temperature()
48    hs_mod.set_and_wait_for_shake_speed(500)
49    protocol.delay(minutes=60) # Let the reactions run for 1 hour
50
51    # Deactivate heater and shaker
52    hs_mod.deactivate_heater()
53    hs_mod.deactivate_shaker()
54    hs_mod.open_labware_latch()
55
56    # Perform any additional steps if necessary

```

Method metadata

Preparing labware and modules

Reagent transfer for Sonogashira reaction

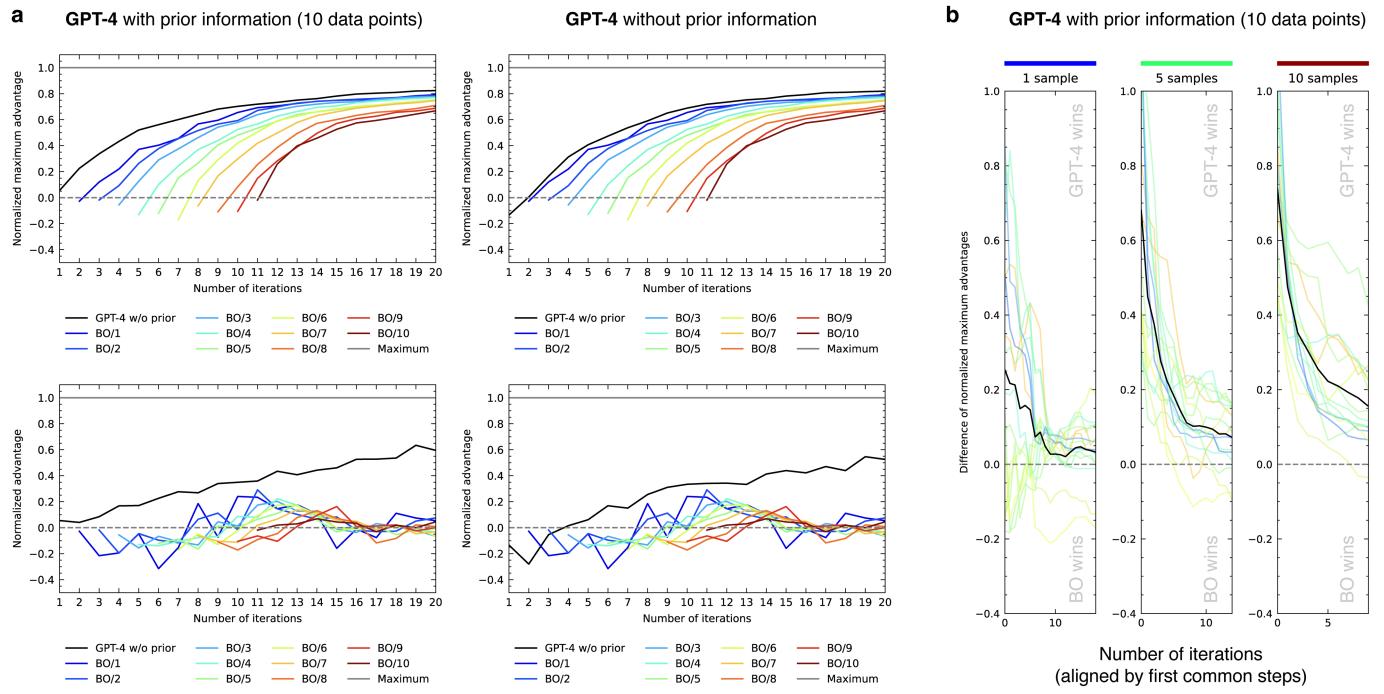
Setting up heater-shaker module

Turning the module off

Extended Data Fig. 2 | Code, generated by Coscientist. The generated code can be split into the following steps: defining metadata for the method, loading labware modules, setting up the liquid handler, performing required reagent

transfers, setting up the heater-shaker module, running the reaction, and turning the module off.

Article



Extended Data Fig. 3 | Additional results on comparison with Bayesian optimization. **a**, GPT-4 models compared with Bayesian optimization performed starting with different number of initial samples. **b**, Compound-by-compound comparison of differences between advantages.