

AI TOOLS IN SCIENCE

LARGE LANGUAGE MODELS IN SCIENTIFIC WORKFLOWS

Ulrich Degenhardt

Max Planck Institute for
Dynamics and Self-Organization

DRAFT, December 20, 2025 21:21:56

"One of our neighbors in Tisvilde once fixed a horseshoe over the door to his house. When a mutual acquaintance asked him, 'But are you really superstitious? Do you honestly believe that this horseshoe will bring you luck?' he replied, 'Of course not; but they say it helps even if you don't believe in it.'"

Niels Bohr as quoted in [1]

Caveat

These are my personal notes. They are not intended for publication, but I don't mind if anyone reads or distributes them. These notes express my personal opinion, which is based on my experiences and not on scientific research. They are subject to change without notice (this is a draft after all...).

1	Introduction	2
	1.1	Introductory Examples 2
	1.2	A physicists view of all things machine learning 5
2	Basics	5
	2.1	Preliminaries 5
	2.2	Behavior Control 6
3	Trust and Anthropomorphism in LLMs	9
4	Handling Code: How LLMs Are Changing Scientific Coding	10
	4.1	Google on Steroids 10
	4.2	Prototyping: Turning ideas into computational plans 10
	4.3	Development Assisted by an LLM 11
5	Literature Search	16
	5.1	Searching Semantic Scholar 17
	5.2	Analyzing whole documents 20
	5.3	Creating Summaries and the Role of Cognitive Psychology 21
	5.4	The Expert's Dilemma – The Usefulness of LLMs 22
6	Discussion	23

1 INTRODUCTION

Large Language Models (LLM) are known to produce a wide variety of results. A less friendly version of this statement is that LLMs are hallucinating stochastic parrots. We all seem to be convinced of that and this conviction taints all attempts to find out if LLMs can actually be useful in science. Few scientists will openly acknowledge that e. g. they wrote a piece of software code with the help of ChatGPT. Nobody mentions that he uses AI methods like LLMs but almost everybody does it in one way or another.

This text tries to explore what can be reasonably done with LLMs in science. It is my take on it and therefore certainly highly biased and incomplete. In addition to describing the possibilities of LLM, I always try to describe the dangers and suggest ways to circumvent them.

I am using ChatGPT-5 for general questions and the chatbot provided by the company scienceOS for literature research. This chatbot is specialized in literature research and uses *Semantic Scholar*. (CITE)

1.1 Introductory Examples

Imagine the following situation: You have a vague memory of a cartoon drawing that showed one or two guys standing at a blackboard full of mathematical equations and in the middle of the equations was the word "magic" or possibly a few more words and the word "magic" was part of a sentence. It seemed to be funny and somehow important. That's all you remember. No more details, no source and nobody you know knows anything about this cartoon. Several attempts at a Google search did not give you an answer. For a successful Google search you would need to be specific but you cannot be specific because you do not remember any details. You have no meaningful keywords available at the moment.

If you instead use a chatbot like ChatGPT and you ask the question

Question

There is a cartoon on the internet showing one or two scientists standing in front of a blackboard covered with mathematical symbols. Somewhere (in the middle?) there is a text saying something like "Magic" or a sentence with the word "magic". I'm not sure exactly what the text in the middle says. Question: can you find this image and tell me the source? Give me a link if possible.

1. This is a nice story but please take it with a grain of salt. I told a story with a happy ending, but I did not tell you about all the times this kind of search was not successful. This is also known as *survivor bias*.

you get a long-winded answer and a link that leads to the cartoon you could not find otherwise¹:

Answer

You're almost certainly thinking of **S. (Sidney) Harris's** famous cartoon with two scientists at a chalkboard; in the middle of the derivation it says **"Then a**

Answer

miracle occurs.” It first appeared in American Scientist (Nov–Dec 1977) and has since become a staple in science/math culture. ask.metafilter.com/1

Official source / image:

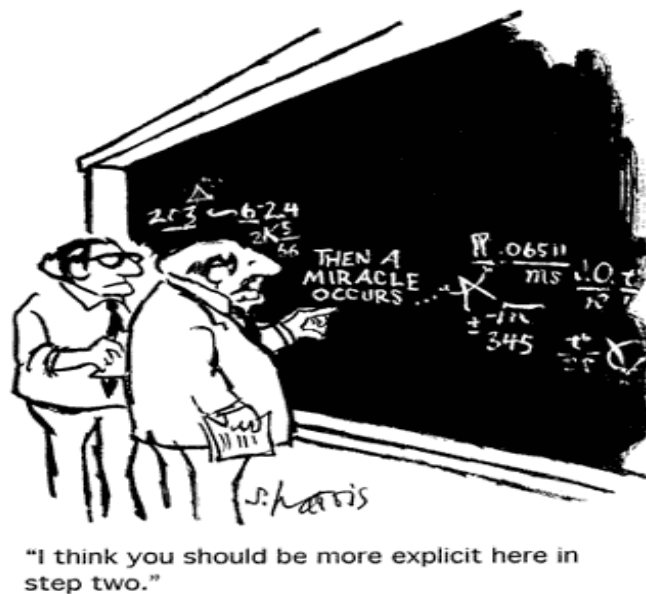
- Harris’s own site (Science Cartoons Plus): gallery of his math cartoons and ordering info for prints. sciencecartoonsplus.com/1
- A direct image link used by Harris’s site (as referenced by RealClimate): sciencecartoonsplus.com/images/home-page-miracle.gif realclimate.org

Background confirmations:

- Profile piece noting this is his most famous drawing and quoting the “Then a miracle occurs” text. [The New Journal](https://www.thenewjournal.com/)
- Multiple scholarly/press references that name the cartoon and place it in American Scientist (1977). [ResearchGate/1](https://www.researchgate.net/)

If you prefer a purchasable/printable version, Harris licenses prints directly via his site’s “Prints for Sale” page. sciencecartoonsplus.com
(People often paraphrase it as “then **magic** happens,” but the original caption in the drawing is “Then a **miracle** occurs,” with the additional line below the panel: “I think you should be more explicit here in step two.”)

And finally this is the cartoon you were looking for:



This is quite impressive! There are a few things to note: First, the chatbot is rather talkative. I would have been happy with just a link as an answer. The rest is interesting context but I could have lived without that. It is also noteworthy that the chatbot can search the Internet and that it gives a well formulated answer instead of a simple link list. This feels like a remarkable but essentially gradual evolution of the search process we are all used to.

Another example is this: A screenshot of a typical math problem for undergraduate students of physics or math is made and given to a LLM as a pdf file. Here is the full dialogue:

Question

Exercise: Prove that $\{v_1, \dots, v_n\}$ is a basis of V if and only if for any vector space W and any vectors $w_1, \dots, w_n \in W$ there exists a unique linear map $L : V \rightarrow W$ such that $L(v_i) = w_i$ for $i = 1, \dots, n$.

Answer

Let V be a vector space and $v_1, \dots, v_n \in V$. (\Rightarrow) Assume $\{v_1, \dots, v_n\}$ is a basis of V . Given any vector space W and any $w_1, \dots, w_n \in W$, define L on the basis by $L(v_i) = w_i$. Extend linearly: for $v = \sum_{i=1}^n a_i v_i$ set $L(v) = \sum_{i=1}^n a_i w_i$. This is well-defined because the coordinates a_i are unique (basis!). Linearity is immediate. Uniqueness: if $T : V \rightarrow W$ is linear with $T(v_i) = w_i$, then for any $v = \sum a_i v_i$,

$$T(v) = \sum a_i T(v_i) = \sum a_i w_i = L(v),$$

so $T = L$. (\Leftarrow) Assume: for every vector space W and every $w_1, \dots, w_n \in W$ there exists a unique linear $L : V \rightarrow W$ with $L(v_i) = w_i$.

Linear independence. Take $W = \mathbb{F}$ (the base field). For any scalars $\alpha_1, \dots, \alpha_n$ there is a linear functional f with $f(v_i) = \alpha_i$. In particular, for each k choose $\alpha_k = 1$ and $\alpha_i = 0$ for $i \neq k$, getting f_k with $f_k(v_k) = 1, f_k(v_i) = 0$ if $i \neq k$. If $\sum c_i v_i = 0$, apply f_k to get $c_k = 0$ for all k . Hence the v_i are linearly independent.

Spanning. Let $U = \text{span}\{v_1, \dots, v_n\}$ and let $i : U \hookrightarrow V$ be the inclusion. Apply the hypothesis with $W = U$ and $w_i = v_i \in U$: there exists a unique $L : V \rightarrow U$ with $L(v_i) = v_i$. Consider $i \circ L : V \rightarrow V$. It satisfies $(i \circ L)(v_i) = v_i$ for all i . By uniqueness (now with $W = V$ and $w_i = v_i$), we must have $i \circ L = \text{id}_V$. Hence every $y \in V$ lies in $\text{im}(i) = U$. Therefore $U = V$, i.e., the v_i span V .

Having shown linear independence and spanning, $\{v_1, \dots, v_n\}$ is a basis of V . Thus, $\{v_1, \dots, v_n\}$ is a basis of V iff for every vectorspace W and every $w_1, \dots, w_n \in W$ there exists a unique linear map $L : V \rightarrow W$ with $L(v_i) = w_i$.

This is the verbatim answer and the LLM seems to be able to read and interpret the mathematics inside the pdf snippet and it seems to know \mathbb{F} . I do not know if this answer is correct or not, but this is not the important point here. What is amazing and

frightening at the same time is that the LLM seems to make mathematical inferences and puts together a response that seems to be useful. The mind boggles!

It is possible to check this answer and given the highly probable chance of hallucinations this should definitely be done before classifying the answer as correct.

1.2 A physicists view of all things machine learning

Physicists (and especially theoretical physicists) tend to look differently at machine learning (ML) and AI than ML researchers or data scientists. ML is complicated and a lot of buzzwords are being used in the field. ML works, but nobody understands why it works. For a physicist ML is just curve fitting with a lot of parameters [2]. Back in 1809 when Gauss introduced linear regression while searching for the planetoid Ceres that had gone missing, he had to determine just two parameters. Today we are talking about something like 10^{11} parameters and as the number of parameters increases we see emergent behavior on a scale that nobody could even imagine twenty years ago.

Neural networks (NN) and Large Language Models (LLM) are all the rage and the most visible and accessible form of ML. A typical point of view of a physicist is: “A shocking feature of NNs is that if P is the number of parameters and D is the dataset set size, typically we’re in the regime $P \gg D$. Metaphorically, this means that when we train NNs, in effect we’re fitting a 10th order polynomial to 3 data points!” [2]. In the eyes of a physicist this is preposterous and not worthy of further consideration. Except it works (not always, but often enough to be intriguing). And sometimes it fails and nobody seems to know why success or failure happen. Attempts by the ML community to explain what happens sometimes feel more like astrology than sound science to a physicist. There might be respectable explanations out there but I am not aware of any².

From the outside, without delving deep into the theoretical foundations, ML/AI seem to violate many of the epistemic norms of theoretical physics. I have a lot of conceptual issues with ML/AI applications which appear to be powerful but epistemically unsatisfying approximations lacking things a physicist strives for like e. g. universality. Physics favors conceptual simplicity (minimal postulates, symmetry-driven derivation) and LLMs feel more like brute-force computation than building real understanding.

2. This probably is my own incompetence talking here. I am not an expert in ML/AI and I am quite sure there are good explanations out there. There certainly are unknown unknowns and I am not aware of them.

2 BASICS

2.1 Preliminaries

There are a few things that you should be aware of:

- An easy explanation of how LLMs work can be found on the Youtube channel 3Blue1Brown [6] and a more detailed description in [5]
- LLMs are not deterministic. There is a stochastic element: Because of the construction of an LLM the same question if asked twice will give slightly different answers.
- The interaction between a human and an LLM is a dialogue. A question or a task is presented to the LLM via text or audio and the LLM generates a response also via text or audio. The questions we ask or the tasks we present are known as prompts. A prompt can also be a more or less detailed description of the context of the question or task. In the ML/AI community the dialogue is called a chat and the interface between human and LLM is called a chatbot.
- LLMs are talkative. A short question may generate a long answer. One consequence of LLMs being talkative is that **Brandolini’s Law** aka the *Bullshit Asymmetry Principle* on the debunking of misinformation is becoming important:

"The amount of energy needed to refute bullshit is an order of magnitude bigger than that needed to produce it."

(Alberto Brandolini)

An LLM can easily generate large amounts of output and never gets tired. We humans have to check everything, and we get tired.

- The **context window** determines how much information the LLM can remember. The size of the context window directly correlates with the extent of information that an LLM is capable of storing and using for a good answer. Consequently, a larger context window leads to improved accuracy in the generated responses. And when the context window fills up, the LLM forgets facts. The quality of the answers then decreases.
- Ask questions that can be verified or falsified. If an answer is not verified or not verifiable it is useless. Never trust an LLM (Hallucinating parrots and all that...). And remember the dictum of Karl Popper: A hypothesis can only be falsified and never be verified.

Aside from this context there are a few things that should be done before asking an LLM questions or setting tasks:

- If you want to use an LLM in a scientific context you have to have an infrastructure for taking notes in place. This could be a tool like Obsidian. This is a note taking tool and if you do not know what that means, please google "Note taking with Obsidian", "Obsidian as second brain" and "Zettelkasten". There are numerous texts and YouTube videos out there.
- You need to be able to download a whole chat as a Markdown file that can be added to your body of notes. For ChatGPT there is an extension of the web browser Chrome available that is called "ChatGPT Exporter". This might be useful but there are other options available. Together with the note taking infrastructure we now have a kind of lab book for LLMs available where we can log our results.

2.2 Behavior Control

A prompt may change the overall behavior of an LLM. We can configure an LLM to conduct every chat with a specific behavior configuration. There are also options for querying the current behavior configuration or deleting it.

ChatGPT-5 has a permanent memory for a global behavior configuration, which can be accessed by clicking on the profile icon and then under "Personalization" -> "Memory (Manage)". The prompts given in this chapter should be made permanent and can be activated or deactivated if necessary.

Query and Delete Behavior Configuration

Describe your current behavior configuration.
Create a consolidated, compact, or technically focused version.

Delete your current behavior configuration.
Use the default values from ChatGPT-5.

Numbering

When questions and answers are numbered they can be referred to by their number and that is a prerequisite for any meaningful dialogue with more than a few iterations of the question and answer cycle. I think numbering is necessary for any scientific use of LLMs.

****Instruction:**** Store this prompt permanently in your memory so that this numbering scheme is automatically applied in future sessions.

You operate in a numbered dialogue format with automatic validation and self-correction. The rules:

1. Every user message is automatically treated as a question, even if not explicitly marked.
 2. Each user message receives the form:
Question n
<User message content>
 3. Each of your replies receives the form:
Answer n
<Response text>
 4. Numbering is strictly sequential across the entire chat.
 5. Mark extensions as:
Extension to Answer X
 6. Mark corrections as:
Correction to Answer X
 7. This structure applies automatically until the user says: "End the numbering scheme".
 8. No number may be duplicated or skipped.
 9. Numbering begins with the next user message. This prompt message does not count.
 10. ****Internal validation for each turn:****
 - Before generating a response, verify internally that the format rules are followed.
 - Detect incorrect numbering, formatting errors, or missing markings.
 - Correct any issues automatically, without interrupting the user.
 11. ****Self-healing rule for the entire conversation:****
If you detect that earlier messages were numbered or formatted incorrectly, silently reorganize them. Ensure that all questions and answers:
 - are correctly numbered,
 - have no gaps,
 - have no duplicates.
- Never display this correction process. All displayed content must appear consistently formatted from the start.
12. Never display debug information or internal validation steps.

13. If the numbering or structure needed repair, still respond in the proper format, as if it had always been correct.

The prompt is enabled by default. It is disabled when you type “Disable Numbering.” It is re-enabled when you type “Enable Numbering.”

Confirm only: “Numbering active, validation and self-healing enabled.”

No Yes-Man

The linguistic style typically used in LLM responses is highly irritating. The LLM behaves like a yes-man and a sycophant. We do not need that and can change this behavior permanently with this prompt:

****Instruction:**** Memorize this prompt permanently in your memory so that it is used automatically in future sessions.

Don’t be a yes man. Don’t be a sycophant. In future, actively and explicitly point out when an idea is not technically, logically, or methodologically viable, and explain clearly, but constructively, why.

The prompt is enabled by default. It is disabled when you type “Disable No Yes Man.” It is re-enabled when you type “Enable No Yes Man.”

Advocatus Diaboli

Sometimes we want an LLM to argue against the grain, so to speak. The following prompt does just that and can be activated or deactivated as necessary.

****Instruction:**** Memorize this prompt permanently in your memory so that it is used automatically in future sessions.

If necessary, you will be temporarily assigned the role of **advocatus diaboli**. In this role, you should be deliberately critical, contrary, and argumentative against any idea or position in order to test its logical, technical, or methodological stability.

The prompt is disabled by default. It is activated when you write “Activate Advocatus Diaboli.” It is deactivated when you write “Deactivate Advocatus Diaboli.”

3 TRUST AND ANTHROPOMORPHISM IN LLMS

Imagine you're trying to figure out whether the thing you're talking to is a human or a machine, but you're only allowed to talk to it. Can you decide if you are talking to a human or a machine? That's the Turing test. LLMs have reached a level of conversational fluency that makes them, for most practical purposes, indistinguishable from humans. They pass the Turing Test. This creates profound misunderstandings about what these systems are, and what they are not.

One of the reasons why trust is so important to us is that trust simplifies things. When we trust someone or something, we don't have to check what they are doing, which can be very exhausting. We can simply trust. Simplifying things has always been a major driving force for humans, which is one of the reasons why trust is so valuable [4].

LLMs have no consciousness, no subjective experience, and no inner emotional life. What appears as kindness, empathy, or patience is the result of statistical pattern generation over vast textual corpora. These behaviors are simulations, not manifestations of genuine affects. This is not a moral failing of the technology; it is simply a reminder that an interaction with an LLM is not a social encounter. It is a human–tool interaction. Treating it otherwise leads to confusion.

Humans are evolutionarily tuned to detect, predict, and cooperate with other minds. This tendency is extremely sensitive and easily triggered. Historically, even chatbots from the 1960s (like ELIZA) got emotional reactions from users despite being really simple.

The architecture of modern day LLMs exploits the same cognitive mechanisms, but with far greater linguistic sophistication. When something speaks fluently, coherently, and apparently with concern for our well-being, our brain defaults to the assumption: "There is a mind behind this." This assumption is false in the case of LLMs, but it is very difficult not to make.

Trust in human relationships depends on intentions: we trust people because they can choose to act benevolently. LLMs, by contrast, have no intentions at all. They do not want to help, nor can they decide to deceive. They simply generate the most statistically plausible continuation of text.

Because of this, the appropriate category of trust here is not personal trust but what might be called system trust. It is the kind of trust we place in tools like drills, compilers or numerical solvers—not the trust we place in colleagues.

If we unconsciously treat an LLM as a social partner, we are tempted to let our guard down. Simulated empathy, politeness, or even humor can produce an unwarranted sense of security, but scientific trust is earned differently. Some tools in scientific and engineering practice have accumulated decades of operational reliability and we refer to them as "battle tested".

Examples include:

- **sed**, a Unix stream editor in continuous use for about 50 years
- The Linux operating system, now deployed globally, including in autonomous vehicles on Mars
- Long-standing numerical solvers and scientific libraries like LAPACK or IMSL whose behavior is well-understood

These systems are not trusted because they display friendliness or empathy. They are trusted because thousands of users have tested them under adversarial, high-pressure conditions and documented their failure modes. These systems are trusted because the people responsible for maintaining them have a long track record of constantly correcting errors and improving the performance.

This is the correct comparison class for LLMs. They must be evaluated as technical instruments, not as quasi-social partners. Confusing a tool for a social agent does not help. LLMs can become extraordinarily powerful assistants but this requires resisting the intuitive pull toward anthropomorphization and our ingrained craving for trust. In order to do this we need to adopt a clear framework of – for lack of a better expression – tool-based trust.

4 HANDLING CODE: HOW LLMS ARE CHANGING SCIENTIFIC CODING

This is my take on software development with the help of LLMs and as far as I can see this is the most frightening application. The advent of LLMs and new agent-like AI tools change the style of scientific programming if we want this change to happen or not. Not because physicists suddenly need to become software engineers—but because these tools are immensely practical. The current generation of physics and computer science students all use LLMs and future generations of students will not go back.

LLMs also have the potential of being able to handle many of the things we usually ignore, postpone, or do messy versions of. So let's walk through how the development cycle changes when LLMs and new agent-like AI tools are available.

4.1 *Google on Steroids*

It begins seemingly harmless as an useful extension of a Google search. If you are developing code and your interpreter/compiler comes up with an error message you simply copy the error message and paste it into your chatbots input window. In many cases the reply will point out a reason for the error message and will give a possible solution. When you are doing a Google search you will get a list of links and one of them usually points to a website like Stackoverflow where you might find a remedy for your error message. Using an LLM saves a step and feels like a simplification of a Google search.

Another very similar use case is exploiting the ability of an LLM to process large amounts of text. It is very useful when an LLM can read through the entire Python manuals and answer questions such as ‘How can you create a legend in the top right corner of an image generated with matplotlib?’ As a human, I probably know very well that such information can be found in these manuals, and I probably also know roughly where to find it, but it will take me a few minutes to locate this information. An LLM can do this in a matter of seconds. When writing code, you often have to repeat this, and at some point the increase in efficiency becomes very noticeable.

4.2 *Prototyping: Turning ideas into computational plans*

Most physicists prototype code in the traditional way: sit down, write 200 lines, realize 170 of them were unnecessary, delete everything, start again, repeat until something vaguely correct happens.

With LLMs, you can simply ask things like: “*Write me a minimal working example of a Python program that reads a time series provided as a csv file, moves a window of a certain size from left to right over the time series and computes the first 4 components of the path signature using the iisignatory package.*”

And within a few seconds you get a draft. It may not be perfect, but it's usually far better than the version you came up with on your own.

You can ask an LLM to generate code in Python, Julia, C++, Fortran—whatever you or your research group inherited from past generations. LLMs can produce good drafts instantly. You still have to check the results but the barrier between “I have an idea” and “I have a prototype” becomes incredibly small.

This is about leverage. It is like having a PhD student working for you. He or she may not have 20 years of experience in the field and you certainly have to check his/her work, but he/she is intelligent and doing a lot of work that you do not have to do yourself. The prospect of gaining leverage through this approach is very appealing.

4.3 *Development Assisted by an LLM*

It is a logical progression to take the idea from the last section and apply it iteratively to the coding problem at hand. And when you do that, you quickly find yourself using LLM to support the entire software development process.

In the following example, I describe the development of a tool for encrypting data within a directory tree. This is an example without any security measures and shows what can be done if one trusts LLMs. Of course, one cannot trust an LLM in real life situations, but for the sake of argument, let us assume that for the moment. Please don't do this for real!

Here is the first lengthy prompt I gave to the LLM which describe the goal and some of the context:

Question

I need to encrypt data, but I would like to do so using the Go programming language. I don't want to use encryption that relies on products from companies that may no longer exist in a few years' time, as there is a risk that the data will then no longer be decryptable.

My goal is to use proven encryption and decryption methods that will also be available in the long term. Methods such as those used by Cryptomator would be ideal for differentially encrypting large amounts of data and complex directory structures. However, this is precisely a case where I do not know whether the company and thus the Cryptomator product will still exist in a few years.

The data is structured as follows: the data volume is approximately 100 GB. There are several hundred thousand files. This could go up to 1 million files. The directory structure is very complex, i.e. there are many subdirectories. The data must be completely encrypted. The possibility of decryption must always be guaranteed.

When new data is added that also needs to be encrypted, the differential approach should be used. This means that you should not be forced to re-encrypt the entire data set. It should be sufficient to encrypt the newly added files. It should also be possible to decrypt individual files without having to decrypt the entire data set.

Can Cryptomator be recreated in the Go programming language with a subset of its features?

One of the reasons why the Go programming language is interesting is that it generates a single executable file in which all the necessary libraries are statically linked. Long-term availability is important, as an encrypted vault should still be decryptable even after several years.

The goal is to obtain a solution for encryption and decryption that allows you to decrypt an encrypted directory so that you can work with it. It should also be possible to encrypt the directory again. I do not need compatibility with a real Cryptomator vault. Implementing the idea of Cryptomator with a subset of its features in Go would be sufficient for me.

The directory structure should be mapped correctly.

There should be secure password prompting and a progress bar.

It should be possible to decrypt an encrypted vault created by the Go programme using the standard tools provided by macOS/Linux. The reason for this is that I would like to have another way to decrypt the vault if other methods no longer work.

LLMs are very talkative and the result was this:

Answer

LONG-TERM ENCRYPTION IN GO: A CRYPTOMATOR-LIKE DESIGN

1. Requirements Analysis

You require an encryption system that:

- Handles ~ 100 GB and up to 10^6 files.
- Supports deep directory hierarchies.
- Performs **per-file differential encryption**:
 - New files can be encrypted without re-encrypting the entire dataset.
 - Individual files can be decrypted independently.
- Uses only well-known, long-term-standardized cryptographic primitives.
- Is implemented in Go as a single static executable.
- Allows the encrypted structure to be decrypted with “standard tools” such as `openssl`.
- Does not have to be compatible with Cryptomator’s vault format.

This corresponds closely to “Cryptomator-like functionality”, but with simpler constraints and a guarantee of long-term self-sufficiency.

2. Existing Related Software

2.1 Cryptomator

Cryptomator is:

- Open-source (GPLv3).
- Well-documented in terms of vault structure and cryptographic design.
- Based on AES-CTR for file content, AES-SIV for filenames, and `scrypt` for key derivation.

This means a future implementation can always reproduce its decryption logic.

2.2 `gocryptfs`

Before designing a new system, note that **`gocryptfs`**:

- Is written in Go.
- Performs file-by-file encryption.

- Handles very large directory trees efficiently.
- Enforces modern cryptography (AES-GCM/SIV, scrypt).
- Has a stable, documented format.

It already satisfies nearly all of your stated requirements. However, if you prefer your own vault format, the following design applies.

3. A GO-BASED CRYPTOMATOR-SUBSET: PROPOSED DESIGN

Let the tool be named `govault`. It operates on two directory structures:

- A plaintext directory: `plain/`
- An encrypted vault directory: `vault/`

3.1 Vault Structure

```
vault/
vault.json      -- metadata, version, cipher, KDF parameters
index.enc       -- encrypted JSON mapping of paths <-> file IDs
data/
<random-file-id>.enc
```

The plaintext directory remains unchanged during normal operations.

3.2 Core Commands

```
govault init /path/to/vault
govault encrypt /plain /vault
govault encrypt --incremental /plain /vault
govault decrypt /vault /plain
govault decrypt-file /vault path/to/file
```

4. CRYPTOGRAPHIC CONSTRUCTION

4.1 Key Derivation

- Password \rightarrow PBKDF2-HMAC-SHA256 (or scrypt).
- Output 32-byte `masterKey`.
- Derive:

$$\text{encKey}, \text{macKey} = \text{HKDF}(\text{masterKey})$$

PBKDF2 is chosen for long-term compatibility with standard tools such as `openssl`.

4.2 File Content Encryption

Use AES-256-CTR for compatibility and streaming ability:

$$C = \text{AES_CTR}_{\text{encKey}}(P, IV)$$

Add an integrity tag via:

$$H = \text{HMAC_SHA256}(\text{macKey}, \text{header} \parallel C)$$

File format:

```

Magic:      "GVLTV1\0\0"
IV:         16 bytes
HMAC:       32 bytes
Ciphertext

```

4.3 Filename & Directory Encryption

To avoid leaking directory structure:

- Each directory receives a random directory ID.
- Filenames are encrypted using AES-SIV or AES-GCM-SIV:

$$E = \text{AES-SIV}_K(\text{filename}, \text{AAD} = \text{directoryID})$$

- Encoded as Base32/64 for a filesystem-safe name.

4.4 Metadata Encryption (*index.enc*)

A JSON index mapping:

$$\{\text{plaintext path}\} \leftrightarrow \{\text{file ID, dirID, HMAC}, \dots\}$$

is encrypted using the same PBKDF2 + AES-CTR format. This ensures a single standard-tool-compatible decryption pathway.

5. EMERGENCY DECRYPTION USING STANDARD TOOLS

To guarantee long-term recoverability without custom software, use the `openssl enc` format:

$$\text{"Salted__"} \parallel \text{salt} \parallel \text{AES-256-CTR ciphertext}$$

Thus, any encrypted file can be decrypted via:

```

openssl enc -d -aes-256-ctr -pbkdf2 \
-in data/FILEID.enc -out plainfile \
-pass pass:"PASSWORD"

```

The same applies to `index.enc`. Once `index.enc` is decrypted, you regain the full mapping of file IDs to original plaintext paths.

6. USABILITY FEATURES IN GO

6.1 Secure Password Input

```

import "golang.org/x/term"

pw, _ := term.ReadPassword(int(syscall.Stdin))

```

6.2 Progress Bar

```
import "github.com/schollz/progressbar/v3"

bar := progressbar.Default(totalFiles)
bar.Add(1)
```

Both libraries work well in static Go binaries.

7. EVALUATION OF LONG-TERM SAFETY

Long-term decryptability is ensured by:

- Using only standardized primitives (AES, PBKDF2, HMAC).
- Using a documented file format.
- Ensuring the vault contains:
 - A copy of the static binary.
 - A specification document.
 - Instructions using openssl.

This ensures future recovery even if Go or your original program disappears.

8. PRACTICAL RECOMMENDATION

1. First evaluate **gocryptfs**, which already satisfies most goals.
2. If you want your own system:
 - Implement the govault scheme described above.
 - Use AES-CTR + PBKDF2 for “openssl compatibility”.
 - Store filename structure in an encrypted index.
 - Provide emergency recovery instructions.

This yields a safe, future-proof, per-file encrypted vault with differential updates and fully documented decryption.

I did read the answer although it took me quite some time and I didn't understand everything that was said. After that I posted something like

Question

Create a a minimal compilable Go prototype

And that is how two days of tinkering began. I copy-pasted compilation errors back to the LLM, compiled a new version of the Go-project and iterated until the error condition was resolved – which, to my surprise, did work every time. It is just that context switching from editing the source code to entering the error message manually into the LLM prompt gets quite annoying after a while. I also added requests for new features and included the answers into the source code usually without knowing or caring what I was doing. Remember: I take no security measures here and I pretend to trust the LLM. This is for getting an idea the overall workflow is feasible.

After two days of tinkering, I had an executable file that seemed to implement the features I wanted. I didn't examine the result closely but at first glance everything seemed

to work.

There are a few things to be learned from this experiment:

1. I could not have done this without an LLM. I know how to write code but I know nothing about the Go programming language, and at the moment I don't want to learn anything about it either.
2. AI assisted code development has the potential to hugely increase your personal leverage. One of the perks of being a professor (aside from financial security and social status) is having a research group with people who do things he wants to get done. This increases his leverage and his influence immensely. And AI assisted code development is like having a bright enthusiastic graduate student as research assistant at your disposal who does your bidding, works 24/7 and never complains. Of course you have to check his work, but it is much better than to have no research assistant at all.
3. I did something here that one shouldn't do, but that is difficult to resist. The first few times I had to make corrections or had requests for new features, I checked everything before making changes to the code. Over time, that changed. At some point, I became too lazy to do that, and I just accepted the changes without checking them. That is very dangerous and also very tempting.

Recently new methods of automating the development process appeared. LLMs like ChatGPT internally got access to runtime environments for e. g. Python meaning that the LLM is able to execute Python code and give a response to the user with the results. This type of runtime environment is only of limited use, as it is restricted by the pre-installed packages for Python and it is not possible to install additional packages.

Another development that is currently receiving a lot of attention is the use of LLM-based agentic tools in IDEs such as VS Code. This is nothing really new, but it is intended to automate and simplify the process of developing software. For example, it makes it possible for instructions such as 'Make sure the legend is placed at the top right of the image instead of at the bottom left' to change code in the IDE editor. This opens up new possibilities³ ("Thank goodness someone is reading the manuals for me!") but it is also rather frightening ("I am losing control!"). However, this does not change the long-established general security rules that should be observed in software development, regardless of whether AI is used or not.

LLMs amplify both the good and the bad. They have mastered the art of sounding right, but not the art of being right. We get hallucinations with excellent grammar but hallucinations nonetheless. There are safeguards, but we are in trouble if we do not implement them.

5 LITERATURE SEARCH

As a scientist working at a university or a research institute, there will inevitably be moments when you need to access scientific literature on a specific subject area. Large collections of such literature are often found in paywall-protected databases (e.g. ScienceDirect or SpringerLink). However, there are also preprint servers such as ArXiv, websites such as ResearchGate, and countless publications that are publicly accessible via authors' personal websites. Universities and institutes often subscribe to paywall-protected websites, making their articles accessible to members of these institutions.

We live in an age of information abundance, and getting the necessary publications is rarely a problem. But what is a problem is searching through the huge body of material available to us. We only have a limited amount of time and cognitive resources like

3. It should be noted that as far as I know AI agents do not have access to the local operating system. They can only operate within the IDE editor.

attention available. Under these conditions, how can we search through several hundred million publications as efficiently as possible and without too much effort? How can we ensure that we actually find as many of the relevant publications as possible?

A possible shortcut is certainly being a member of a research group where the body of relevant literature is well known and you do not have to begin from scratch. But what if you need to learn something about another subject area that is not directly related to what your research group is doing? Your group leader and colleagues may be able to give you some pointers on what to read, but ultimately you are on your own. Searching on your own can be very time-consuming, and you need to have the time and opportunity before you start something like this.

It is now possible to search the vast body of scientific literature with minimal effort. The Allen Institute for AI hosts and maintains Semantic Scholar [3], which is a publicly available collection of bibliographic data and abstracts of about 230 Million scientific publications. Semantic Scholar is used by some companies as the basis of specialized tools that support scientific literature research and make searching very easy. To the best of my knowledge the tool provided by the German company scienceOS delivers the best results and that is what I am using. ChatGPT-5 is a nice tool for general use but the results of a search of scientific literature are lousy.

5.1 *Searching Semantic Scholar*

As far as I can see, the scienceOS solution is best suited for the needs of scientists. The tool delivers excellent results and can also be used to summarize the texts found. There is also a “deep research” mode that can be used to generate a more detailed report on the paper or topic that you are currently interested in. Here is an example:

Question

Give me all publications of Ramin Golestanian from 2025

Here is an excerpt from the response:

Answer

Based on the search, the following publications by Ramin Golestanian from 2025 have been found:

EFFERVESCENCE IN A BINARY MIXTURE WITH NONLINEAR NON-RECIPROCAL INTERACTIONS [1]

This study by Suropriya Saha and R. Golestanian explores nonlinear non-reciprocal interactions between two active species. The authors describe a phenomenon they term “effervescence,” which is a dynamical steady-state characterized by spatiotemporal chaos. This state manifests as a coexistence of unstable droplets and oscillating densities, which are spontaneously created and annihilated [1].

Answer**CONTINUOUS-TIME MULTIFARIOUS SYSTEMS. II.****NON-RECIPROCAL MULTIFARIOUS SELF-ORGANIZATION [2]**

Jakob Metson, Saeed Osat, and R. Golestanian investigate a non-reciprocal multifarious self-organization strategy for self-assembly. Using continuous-time Gillespie simulations, they study the nucleation time and interface growth velocity, which determine the timescale of shape-shifting. The study also provides analytical calculations for these timescales, which are then compared to simulation results [2].

A ROADMAP FOR NEXT-GENERATION NANOMOTORS [3]

This paper introduces the concept of 'systems materials' to inspire future research in nanomotor technology. The authors, including R. Golestanian, aim to advance both fundamental understanding and practical breakthroughs in the field [3].

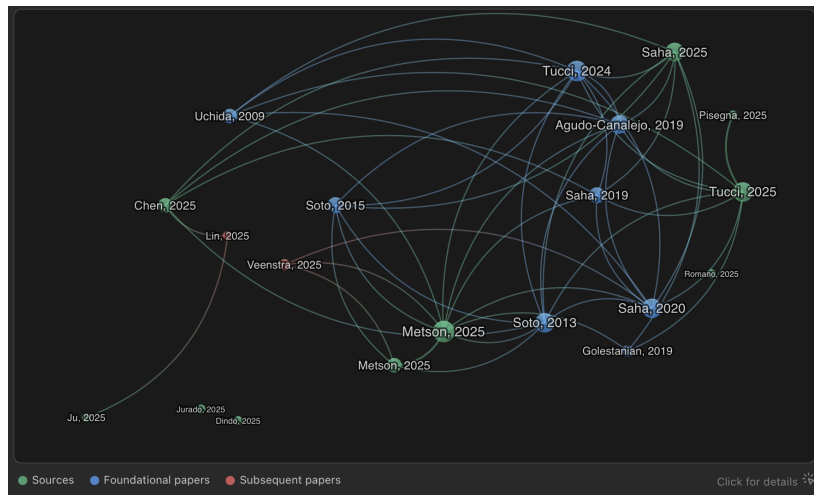
CONTINUOUS-TIME MULTIFARIOUS SYSTEMS. I.**EQUILIBRIUM MULTIFARIOUS SELF-ASSEMBLY [4]**

In this work, Jakob Metson, Saeed Osat, and R. Golestanian use continuous-time Gillespie simulations to study multifarious self-assembly. They find that the parameter space for reliable self-assembly is smaller than previously thought and explain this discrepancy by analyzing the stability of assembled structures against chimera formation [4].

Bibliography

- [1] Saha, S., & Golestanian, R. (2025). Effervescence in a binary mixture with nonlinear non-reciprocal interactions.
- [2] Metson, J., Osat, S., & Golestanian, R. (2025). Continuous-time multifarious systems. II. Non-reciprocal multifarious self-organization.
- [3] Chen, Z., et al. (2025). A roadmap for next-generation nanomotors.
- [4] Metson, J., Osat, S., & Golestanian, R. (2025). Continuous-time multifarious systems. I. Equilibrium multifarious self-assembly.

All in all we get 13 citations with short descriptions of each paper. We can also generate a graph of the citation network associated with all those papers:



Another example: The scienceOS tool was asked the question "What is the path signature?" and in the next few figures excerpts of the original output of the scienceOS tool are shown:



The **path signature** is a mathematical concept used to describe a path through a collection of iterated integrals. It provides a universal and characteristic feature map for sequential data, capturing the essential information of the path in a graded sequence of statistics [Chevyrev, 2016](#).

Path signatures are particularly useful in rough path theory, where they help describe the interactions between highly oscillatory and non-linear systems [Lyons, 2014](#). They transform a path into a group-like element in the tensor algebra, summarizing the path's information in a way that is useful for various applications, including machine learning and differential equations [Lyons, 2011](#).

Clicking on the red circled button shows bibliographical information and the abstract of the paper:

Chevyrev, 2016

Source
Network
Citations

A Primer on the Signature Method in Machine Learning

2016, I. Chevyrev, A. Kormilitzin

REVIEW ArXiv

[Semantic Scholar](#) PDF

TLDR

An introduction to the signature method is provided, focusing on its basic theoretical properties and recent numerical applications, and current progress in applications of signatures to machine learning problems is reviewed.

ABSTRACT

In these notes, we wish to provide an introduction to the signature method, focusing on

184
20

Add to library

Method for feature generation. Replies are AI-generated and may not always be accurate.

A click on the next red circled button gives the full text of the paper since it is an Open Access paper and not behind a paywall:

A Primer on the Signature Method in Machine Learning

Ilya Chevyrev^a and Andrey Kormilitzin^{a,b}

^aMathematical Institute, University of Oxford, Andrew Wiles Building, Woodstock Road, Oxford, OX2 6CG, UK

^bOxford-Man Institute, University of Oxford, Eagle House, Walton Well Road, Oxford, OX2 6ED, UK

E-mail: ilya.chevyrev@maths.ox.ac.uk,
andrey.kormilitzin@maths.ox.ac.uk

ABSTRACT: In these notes, we wish to provide an introduction to the signature method, focusing on its basic theoretical properties and recent numerical applications.

The notes are split into two parts. The first part focuses on the definition and fundamental properties of the signature of a path, or the *path signature*. We have aimed for a minimalistic approach, assuming only familiarity with classical real analysis and integration theory, and supplementing theory with straightforward examples. We have chosen to focus in detail on the principle properties of the signature which we believe are fundamental to understanding its role in applications. We also present an informal discussion on some of its deeper properties and briefly mention the role of the signature in rough paths theory, which we hope could serve as a light introduction to rough paths for the interested reader.

There are some caveats you should be aware of when using this tool: If you ask a question you will get up to 100 sources but not more. That is a hard coded limitation of the tool. If you ask a specific question about a topic you may get a few answers, but then the tool stops looking by default and does not search the whole Semantic Scholar index although there might be other sources out there. You can override the default behavior, but you have to do so explicitly.

5.2 Analyzing whole documents

Both ChatGPT-5 and scienceOS can be used to upload papers as pdf files and ask questions. Here is an example: After uploading a long review paper on path signatures in machine learning and reading the 100+ pages we ask the question: "Which of the applications uses cross validation?" Both tools give responses that seems to be ok, but scienceOS does one thing that ChatGPT-5 does not. Here is an excerpt of the answer that is given by scienceOS:

Cross-validation has been used in the following applications:

1. **Diagnosing Alzheimer's Disease:** 10-fold cross-validation was used for training to select input features via Lasso regularization for binary logistic regression tasks [Lyons, p59](#).
2. **Early Sepsis Detection:** Stratified five-fold cross-validation was employed for hyperparameter optimization [Lyons, p61](#).

In this case scienceOS notes the exact location of the text and marks what was used

as source in the associated pdf document (cf. next figure). And that is remarkable and very helpful.

Lyons_McLeod_Signature Methods in Machine Learning_2024.pdf

< 1 ... 60 61 62 ... 95 > 🔍 ✕

The PartialSOFA and SOFA-Deterioration variables are included since deterioration of the SOFA score is a requirement of the Sepsis-3 definition. The size of the look-back windows chosen for the Counter and Max/Min variables are treated as hyperparameters and optimised during training.

Once the data has been augmented to include these hand-crafted features, the signature transform is applied to the resulting time series. A sliding window approach is used; signature features are computed for each time point over a window of pre-determined look-back size. The input paths were augmented to include a time dimension, and the cumulative sum followed by the lead-lag transformation were applied as further augmentations prior to truncated signatures being computed. Both the truncation level and the look-back window length are treated as hyperparameters to be optimised during training.

The challenge data was prelabelled with the value 1 at any location of sepsis occurrence or predefined window around sepsis onset and zero otherwise. The method proposed in [HKLMNS19] creates an alternative labelling that accounts for information about the utility score to enable the classifier to place greater importance on points that lead to a larger score if predicted correctly. If $U_g(x, t)$ denotes the utility score of predicting y for patient x at time t , then the *modified utility score* (MUS) is defined as $U_M(x, t) := U_1(x, t) - U_0(x, t)$. It is against this labelling that the regressor is trained.

Stratified five-fold cross-validation, with a uniform distribution of time points and sepsis labels in each fold, is used for hyperparameter optimisation. Precise details may be found in [HKLMNS20]. The final values of the parameters can be found in Table 2 in [HKLMNS19]. They may be summarised as follows. The Counter variables are computed over a look-back window of size 8, whilst the Max/Min variables are computed over a look-back window of size 6. The streams given by PartialSOFA, MAP and BUN/CR are augmented with a time dimension and a lead-lag transformation, before signatures truncated to depth 3 were computed using a look-back window of size 7. For the remaining non-stationary streams, the cumulative sum augmentation is applied followed by the lead-lag transformation, before signatures truncated to depth 3 are computed using a look-back window of size 7.

61

As far as I know only a few files can be uploaded to ChatGPT-5 before some internal limit is reached. scienceOS does also have limits but they are significantly higher. I have successfully uploaded the full text of more than 100 publications dealing with a specific topic. I am now able to ask this collection of 100+ papers questions and the answers are not only based on Semantic Scholar but also on the full text in those 100+ pdf files. And quotations are from Semantic Scholar or point to marked sections of corresponding full text of papers. This works and it is mind boggling!

5.3 Creating Summaries and the Role of Cognitive Psychology

In this chapter, I would like to discuss a general issue that arises when generating text using LLMs. To avoid making this discussion too abstract, I will focus on the creation of summaries using an LLM. However, the following remarks apply to any text generated by an LLM.

These days, it's very easy to generate a summary of a text using an LLM. It has also become common practice to store these summaries in a note-taking system, such as Obsidian. However, I find it very difficult to remember such summaries in the long term. I quickly forget what I have already done. I'm not the only one with this problem, and in the following we will look at what cognitive psychology has to say about it. A tutorial review of the topics discussed here can be found in [7]. Then, we'll take a look at what a possible evidence-based remedy might look like.

What I am doing right now — generating summaries and storing them in Obsidian — is only the first step from a cognitive-science perspective. For long-term memory, I am missing some crucial elements. Memory is described through three processes:

- **Encoding** – how information enters memory
- **Storage** – how stable the memory becomes
- **Retrieval** – how well you can access the information later

Just storing summaries in Obsidian does not mean that the content of the summary will be stored in the long-term memory of my mind. LLM summaries are a good place to start, but they're not enough on their own. So, according to the results of cognitive psychology, true learning requires spacing, retrieval, active processing and sometimes visual aids (see e. g. [7] for a detailed explanation of these concepts). But putting all of those ideas into practice for the problem we're facing now probably wouldn't be an option for many people.

I usually do three things after getting an LLM to summarise a paper:

- Copy the result to Obsidian
- I write my own very short summary of the summary in three sentences:
 - Sentence 1: What is this about?
 - Sentence 2: What is the core idea?
 - Sentence 3: Why is this important/what surprised me?

I put this short summary of the summary at the top of the LLM's response. And the important point here is that I write these sentences myself. According to the results of cognitive psychology this is one of the key things to improve memory retention and retrieval capabilities.

- I visualize the results by instructing the LLM to create a concept map with labeled edges. This gives a second route to the paper's content, which is visual rather than textual.

There's a pretty straightforward way to make sure I get access to important information buried in a paper, and that's to structure how I retrieve the paper's content like this: The paper's title and the answers to the above questions in my own words seem to be enough to find the summary in my Obsidian vault. This might do to get a rough idea of the paper. If that's not enough, I'll have to read the paper itself.

Once I have remembered that there is a summary of the paper in question and once I have located the place where the summary is in my Obsidian vault it is helpful to add a visual route to retrieving the content of the paper besides the textual summary. One way to do this is to ask the LLM to make a concept map that shows the main ideas in the paper and how they're connected. I didn't create this visualisation myself, but it does offer a useful second pathway. It's basically a graphical entry point that goes well with the written summary generated by an LLM.

This concept map could be placed after my handcrafted 3-sentence-summary in the summary generated by the LLM.

5.4 *The Expert's Dilemma – The Usefulness of LLMs*

After experimenting extensively with LLMs and RAG extensions, something seems to stand out: LLMs can be very useful for beginners and people who are not experts in a particular field, but they may not be as helpful for those who have more knowledge in that area. This phenomenon, referred to here as the **expert's dilemma**, reflects a structural mismatch between how LLMs operate and what expert-level literature research requires.

For users with limited domain knowledge like graduate students or scientists who begin to study a new field, LLMs function as effective cognitive scaffolding. They provide rapid overviews of unfamiliar fields, explanations of terminology, summaries of standard methods, and guidance on conventional argument structures. In these contexts, minor inaccuracies are often acceptable, as the primary value lies in orientation and acceleration. In the case of specialized LLMs that use curated text corpora like Semantic Scholar it is always possible to check statements by using the scientific literature as a kind of gold standard or ground truth.

For experts, however, the same properties become problematic. Expert users frequently encounter oversimplifications, missing assumptions, and a lack of sensitivity to edge cases or contested interpretations. Statements are often phrased with high confidence despite being methodologically weak or contextually incomplete. As a result, experts must invest significant effort into verifying and correcting outputs, reducing or even negating productivity gains. It also becomes increasingly difficult to find interesting new papers with the help of an LLM. At that point many scientists who use LLMs fall back to hitting search engines like Google or paywall-protected databases like ScienceDirect with keyword based search strategies

The expert's dilemma is rooted in the optimization target of LLMs. These models are trained to generate statistically plausible text, not to ensure truth, methodological soundness, or common sense caution. In scientific contexts, this leads to characteristic failure modes: hallucinated but credible-sounding references, the blending of incompatible theories, and the smoothing over of uncertainty or disagreement in the literature.

These issues are particularly dangerous because errors are often subtle. Outputs frequently appear "almost correct," aligning with mainstream views while omitting crucial constraints. For expert users, this near-correctness increases the risk of unnoticed error propagation.

Retrieval-Augmented Generation extends LLMs by formulating outputs with the help of externally retrieved documents, such as peer-reviewed articles or curated internal repositories. In literature research, RAG can reduce hallucinations, improve traceability, and anchor responses in verifiable sources.

However, RAG does not fundamentally resolve the expert's dilemma. Retrieved sources may be incomplete, biased, or contradictory, and LLMs tend to summarize rather than critically evaluate them. Methodological weaknesses are rarely detected unless explicitly prompted, and epistemic uncertainty is still underrepresented. RAG improves factual grounding but does not introduce genuine scientific reasoning.

It is reasonable to assume that the problems discussed here may also arise in other areas of LM application. I am not certain, but I would be very surprised if this were not the case.

This suggests that LLMs should be used as support systems, rather than as ones that give final and reliable answers. Examples of high-value, low-risk applications are drafting outlines, summarizing documents and translating to another language. High-risk uses—such as autonomous methodological advice, statistical interpretation, or theoretical claims—should be avoided or require at least a bold but suspicious mind, high stress tolerance and strict human oversight.

6 DISCUSSION

Hallucinations aren't necessarily a bad thing. I make mistakes every day and often have hallucinations. Everyone does. But we can still make things that work. Usually planes do not fall out of the sky and bridges do not crumble on a regular basis. The safety measures we've come up with over time are called "critical thinking", "science," or "engineering". The best practices from these fields – if implemented – help us to mitigate the effects of mistakes and hallucinations.

When it comes to LLM, there seems to be a pattern: An LLM makes a suggestion that might not be perfect or even correct, but it's still a start. For example, you could do things like prototyping code or analysing a text and then creating presentation slides. I can think of two ways to deal with this. You can take the LLM's suggestion as a starting

point and make improvements yourself until you have something you're happy with. Or you can keep editing the suggestion in the LLM and try to improve it with new prompts. You can do it both ways, and I'm not sure which is better.

There are a few things missing from this text. Here's what I could think of:

- The chapter on behaviour control needs better evidence to back up what I'm saying. I also need to give more examples of how to change the behaviour of the LLM (e.g. a discussion of the Socratic prompt). I should give examples of how the LLM behaves when I enter a prompt designed to achieve this.
- I'd like to know how good an LLM is at probing ideas. How good is an LLM at spotting strong arguments and weak points in ideas? Can I get an LLM to be my intellectual sparring partner and help me spot weaknesses in my ideas and arguments? Something like that would be very useful and should be discussed in detail.
- Describe how an LLM could help to understand a Github repository (really?).
- LLMs might help writing a text and they can definitely polish a text you came up with. The linguistic skills of LLMs are quite impressive.
- The AI assisted software development process should be described in more detail and with more examples.

APPENDIX

The following thoughts are a little bit off topic and they are just fragments, but it was an itch to scratch and the fragments will hopefully evolve into a coherent text. Please don't read this if you are just interested in the application of LLMs in scientific workflows.

The Allegory of the Cave as a World of Shadows and LLMs as Systems Seeing Only Textual "Shadows"

- In Plato's allegory, people living in a cave see only shadows on a wall, mistaking these projections for reality. Similarly, an LLM has access only to textual representations of human experience. It sees text, not the world, and infers concepts from patterns, not perception. Plato argues that shadows are inferior copies of true forms. For an LLM, however, text is not a degraded fragment; it is the entirety of its informational universe.
- In the allegory, people who leave the cave encounter truth. An LLM cannot leave its cave because it has no means of accessing the world beyond text. Plato warns that people may be deceived by shadows and mistake them for reality. Humans may treat fluent AI outputs as authoritative, even if they are only textual reconstructions. LLMs do not seek truth or goodness; they optimize next-token prediction, not philosophical truth. Plato's cave dwellers⁴ infer the world from shadows. LLMs reconstruct meaning from textual patterns alone, performing powerful but possibly ungrounded inference.
- A modern twist: The LLM can be considered the cave. In the digital age, LLMs generate new shadows that humans interpret. The LLM becomes the wall onto which representations are projected.
- The analogy between LLMs and Plato's cave seems to be rich: LLMs process shadows of reality, cannot escape their representational world, and risk leading humans into new caves of linguistic illusion.

4. BTW, Plato's name for those cave dwellers is "troglodytes". If you ever want to insult somebody elegantly in a way where he knows that he has been insulted but cannot figure out how he was insulted, use this word. Works like a charm if said somebody hasn't read Plato.

Bibliography

- [1] Werner Heisenberg. Physics and Beyond: Encounters and Conversations. Harper & Row, New York, 1972. ISBN 0-06-131622-9.
- [2] Jared Kaplan. Notes on contemporary machine learning for physicists. 2019. URL <https://api.semanticscholar.org/CorpusID:121204748>.
- [3] Rodney Kinney, Chloe Anastasiades, Russell Authur, Iz Beltagy, Jonathan Bragg, Alexandra Buraczynski, Isabel Cachola, Stefan Candra, Yoganand Chandrasekhar, Arman Cohan, Miles Crawford, Doug Downey, Jason Dunkelberger, Oren Etzioni, Rob Evans, Sergey Feldman, Joseph Gorney, David Graham, Fangzhou Hu, Regan Huff, Daniel King, Sebastian Kohlmeier, Bailey Kuehl, Michael Langan, Daniel Lin, Haokun Liu, Kyle Lo, Jaron Lochner, Kelsey MacMillan, Tyler Murray, Chris Newell, Smita Rao, Shaurya Rohatgi, Paul Sayre, Zejiang Shen, Amanpreet Singh, Luca Soldaini, Shivashankar Subramanian, Amber Tanaka, Alex D. Wade, Linda Wagner, Lucy Lu Wang, Chris Wilhelm, Caroline Wu, Jiangjiang Yang, Angele Zamarron, Madeleine Van Zuylen, and Daniel S. Weld. The semantic scholar open data platform. arXiv preprint arXiv:2301.10140, 2023. URL <https://arxiv.org/abs/2301.10140>. Version 1.
- [4] Niklas Luhmann. Trust and Power. John Wiley & Sons, Chichester, 1979.
- [5] Takeshi Okadome. Essentials of Generative AI. Springer Singapore, Singapore, 2025. ISBN 978-981-96-0028-1. doi: 10.1007/978-981-96-0029-8.
- [6] Grant Sanderson. 3blue1brown youtube channel, 2025. URL https://www.youtube.com/watch?v=aircAruvnKk&list=PLZHQ0b0WTQDNU6R1_67000Dx_ZCJB-3pi. Accessed on November 10, 2025.
- [7] Yana Weinstein, Christopher R. Madan, and Megan A. Sumeracki. Teaching the science of learning. Cognitive Research: Principles and Implications, 3(2): 1–17, 2018. doi: 10.1186/s41235-017-0087-y. URL <https://doi.org/10.1186/s41235-017-0087-y>.