

June 2023

29 posts: [5 entries](#), [18 links](#), [6 quotes](#)

June 1, 2023

He notes that one simulated test saw an AI-enabled drone tasked with a SEAD mission to identify and destroy SAM sites, with the final go/no go given by the human. However, having been 'reinforced' in training that destruction of the SAM was the preferred option, the AI then decided that 'no-go' decisions from the human were interfering with its higher mission – killing SAMs – and then attacked the operator in the simulation.

[UPDATE: This turned out to be a "thought experiment" intentionally designed to illustrate how these things could go wrong.]

— [Highlights from the RAeS Future Combat Air & Space Capabilities Summit](#)

[11:07 pm](#) / [ai](#), [ethics](#), [ai-ethics](#)

June 2, 2023

[Vector Search](#). Amjith Ramanujam provides a very thorough tutorial on implementing vector similarity search using SentenceTransformers embeddings (all-MiniLM-L6-v2) executed using sqlite-utils, then served via datasette-sqlite-vss and deployed using Fly.

[5:02 am](#) / [sqlite](#), [ai](#), [datasette](#), [fly](#), [embeddings](#)

June 3, 2023

[pytest-icdiff](#) ([via](#)) This is neat: "pip install pytest-icdiff" provides an instant usability upgrade to the output of failed tests in pytest, especially if the assertions involve comparing larger strings or nested JSON objects.

[4:59 pm](#) / [python](#), [testing](#), [pytest](#)

June 4, 2023

There was an exchange on Twitter a while back where someone said, 'What is artificial intelligence?' And someone else said, 'A poor choice of words in 1954'. And, you know, they're right. I think that if we had chosen a different phrase for it, back in the '50s, we might have avoided a lot of the confusion that we're having now.

— [Ted Chiang](#)

[2:59 pm](#) / [ai](#), [ted-chiang](#)

It's infuriatingly hard to understand how closed models train on their input

One of the most common concerns I see about large language models regards their training data. People are worried that anything they say to ChatGPT could be memorized by it and spat out to other users. People are concerned that anything they store in a private repository on GitHub [might be used as training data](#) for future versions of Copilot.

[... [1,465 words](#)]

6:09 pm / [ai](#), [openai](#), [generative-ai](#), [chatgpt](#), [llms](#), [anthropic](#), [claude](#), [training-data](#)

Weeknotes: Parquet in Datasette Lite, various talks, more LLM hacking

I've fallen a bit behind on my weeknotes. Here's a catchup for the last few weeks.

[... [769 words](#)]

9:14 pm / [projects](#), [speaking](#), [tutorials](#), [datasette](#), [parquet](#), [weeknotes](#), [datasette-lite](#), [llms](#)

June 5, 2023

[Logan Kilpatrick \(OpenAI\)](#). "The API does not just change without us telling you. The models are static there."

That's the official line on the ongoing questions concerning whether OpenAI's models have been degrading in quality over the last few weeks and months.

Worth noting that this mentions the API but doesn't mention ChatGPT itself, which I suspect gets model updates a lot more frequently than the models served through the API.

[3:49 pm](#) / [ai](#), [openai](#), [generative-ai](#), [chatgpt](#), [llms](#), [logan-kilpatrick](#)

If you give feedback that isn't constructive your feedback is worthless. I know that sounds harsh but it is. If you give unconstructive feedback you might as well not be saying anything. If you just look at something and go "That's stupid" or "I don't like that" - that's worthless feedback, nobody can do anything with that. They're not going to start throwing darts against the wall until you say "Oh OK, I like that". You have to say something more.

— [Timothy Cain](#)

[4:58 pm](#) / [communication](#)

June 8, 2023

[ChatGPT Plugins Don't Have PMF](#). Sam Altman was recently quoted (in a since unpublished blog post) noting that ChatGPT plugins have not yet demonstrated product market fit.

This matches my own usage patterns: I use the “browse” and “code interpreter” modes on a daily basis, but I’ve not found any of the third party developer plugins to stick for me yet.

I like Matt Rickard’s observation here: “Chat is not the right UX for plugins. If you know what you want to do, it’s often easier to just do a few clicks on the website. If you don’t, just a chat interface makes it hard to steer the model toward your goal.”

4:59 am / [ai](#), [openai](#), [generative-ai](#), [chatgpt](#), [llms](#), [code-interpreter](#), [sam-altman](#), [coding-agents](#)

[First Impressions of Vision Pro and VisionOS](#). John Gruber’s description of his thirty minute Vision Pro demo includes a bunch of details I haven’t seen described anywhere else, including how calibration and corrective lenses work and how precise and stable the overlays of additional information are.

6:16 am / [apple](#), [john-gruber](#), [vr](#)

[Examples of weird GPT-4 behavior for the string “ davidjl”](#). GPT-4, when told to repeat or otherwise process the string “ davidjl” (note the leading space character), treats it as “jndl” or “jspb” or “JDL” instead. It turns out “ davidjl” has its own single token in the tokenizer: token ID 23282, presumably dating back to the GPT-2 days.

Riley Goodside refers to these as “glitch tokens”.

This token might refer to Reddit user davidjl123 who ranks top of the league for the old /r/counting subreddit, with 163,477 posts there which presumably ended up in older training data.

9:29 am / [reddit](#), [ai](#), [openai](#), [generative-ai](#), [riley-goodside](#), [gpt-4](#), [llms](#), [tokenization](#)

[Understanding GPT tokenizers](#)

GPT token encoder and decoder

Enter text to tokenize it:

The dog eats the apples
El perro come las manzanas
片仮名

464 3290 25365 262 22514 198 9527 583 305 1282 39990 582 15201 292 198 31965
229 20015 106 28938 235
21 tokens



Large language models such as GPT-3/4, LLaMA and PaLM work in terms of tokens. They take text, convert it into tokens (integers), then predict which tokens should come next.

[... [1,575 words](#)]

[8:37 pm](#) / [projects](#), [ai](#), [gpt-3](#), [openai](#), [generative-ai](#), [gpt-4](#), [llms](#), [tokenization](#)

[simpleai](#) [chat](#) ([via](#)) Max Woolf released his own Python package for building against the GPT-3.5 and GPT-4 APIs (and potentially other LLMs in the future).

It's a very clean piece of API design with some useful additional features: there's an AsyncAIChat subclass that works with Python asyncio, and the library includes a mechanism for registering custom functions that can then be called by the LLM as tools.

One trick I haven't seen before: it uses a combination of max_tokens: 1 and a ChatGPT logit_bias to ensure that answers to one of its default prompts are restricted to just numerals between 0 and 9. This is described in the PROMPTS.md file.

[# 9:06 pm](#) / [python](#), [ai](#), [max-woolf](#), [openai](#), [prompt-engineering](#), [generative-ai](#), [chatgpt](#), [llms](#)

[June 12, 2023](#)

Cellphones are the worst thing that's ever happened to movies. It's awful. [...] I think you could talk to a hundred storytellers and they would all tell you the same thing. It's so hard to manufacture drama when everybody can get a hold of everybody all the time. It's just not as fun as in the old days when the phone would ring and you didn't know who was calling.

— [Steven Soderbergh](#)

[# 6:13 pm](#) / [screen-writing](#), [mobile](#)

[June 13, 2023](#)

[OpenAI: Function calling and other API updates](#). Huge set of announcements from OpenAI today. A bunch of price reductions, but the things that most excite me are the new gpt-3.5-turbo-16k model which offers a 16,000 token context limit (4x the existing 3.5 turbo model) at a price of \$0.003 per 1K input tokens and \$0.004 per 1K output tokens—1/10th the price of GPT-4 8k.

The other big new feature: functions! You can now send JSON schema defining one or more functions to GPT 3.5 and GPT-4—those models will then return a blob of JSON describing a function they want you to call (if they determine that one should be called). Your code executes the function and passes the results back to the model to continue the execution flow.

This is effectively an implementation of the ReAct pattern, with models that have been fine-tuned to execute it.

They acknowledge the risk of prompt injection (though not by name) in the post: “We are working to mitigate these and other risks. Developers can protect their applications by only consuming information from trusted tools and by including user confirmation steps before performing actions with real-world impact, such as sending an email, posting online, or making a purchase.”

[# 5:34 pm](#) / [ai](#), [gpt-3](#), [openai](#), [prompt-engineering](#), [prompt-injection](#), [generative-ai](#), [chatgpt](#), [gpt-4](#), [llms](#)

[Llama encoder and decoder](#). I forked my GPT tokenizer Observable notebook to create a similar tool for exploring the tokenization scheme used by the Llama family of LLMs, using the new llama-tokenizer-js JavaScript library.

[# 10:37 pm](#) / [ai](#), [observable](#), [generative-ai](#), [llama](#), [llms](#), [tokenization](#)

June 14, 2023

[Emergency Pod: OpenAI's new Functions API, 75% Price Drop, 4x Context Length](#) ([via](#)) I participated in a Twitter Spaces conversation last night about the new OpenAI functions mechanism. The recording has now been turned into a Latent Space podcast, and swyx has accompanied the recording with a detailed write-up of the different topics we covered.

[# 7:23 pm](#) / [podcasts](#), [speaking](#), [ai](#), [openai](#), [generative-ai](#), [llms](#), [podcast-appearances](#)

[Example of OpenAI function calling API to extract data from LAPD newsroom articles](#) ([via](#)) Fascinating code example from Kyle McDonald. The OpenAI functions mechanism is intended to drive custom function calls, but I hadn't quite appreciated how useful it can be ignoring the function calls entirely. Kyle instead uses it to define a schema for data he wants to extract from a news article, then uses the gpt-3.5-turbo-0613 to get back that exact set of extracted data as JSON.

[# 8:57 pm](#) / [data-journalism](#), [ai](#), [openai](#), [generative-ai](#), [llms](#)

June 15, 2023

[When Zeppelins Ruled The Earth](#) ([via](#)) 15 years ago I put together a talk about the history of Zeppelins which I presented a bunch of different times in various different configurations. As far as I know there are no existing videos of it, but I found an MP3 recording today and decided to splice it together with the slides to create a video of the 6m47s version I gave at the Skillswap on Speed lightning talks event in Brighton on the 28th October 2008.

Notes on how I edited the video together using iMovie in the [via](#) link.

[# 8:16 pm](#) / [my-talks](#), [zeppelins](#)

June 17, 2023

[sqlean.py: Python's sqlite3 with extensions](#). Anton Zhiyanov built a new Python package which bundles a fresh, compiled copy of SQLite with his SQLean family of C extensions built right in. Installing it gets you the latest SQLite—3.42.0—with nearly 200 additional functions, including things like `define()` and `eval()`, `fileio_read()` and `fileio_write()`, `percentile_95()` and `uuid4()` and many more. “`import sqlean as sqlite3`” works as a drop-in replacement for the module from the standard library.

[# 10:42 pm](#) / [python](#), [sqlite](#), [anton-zhiyanov](#)

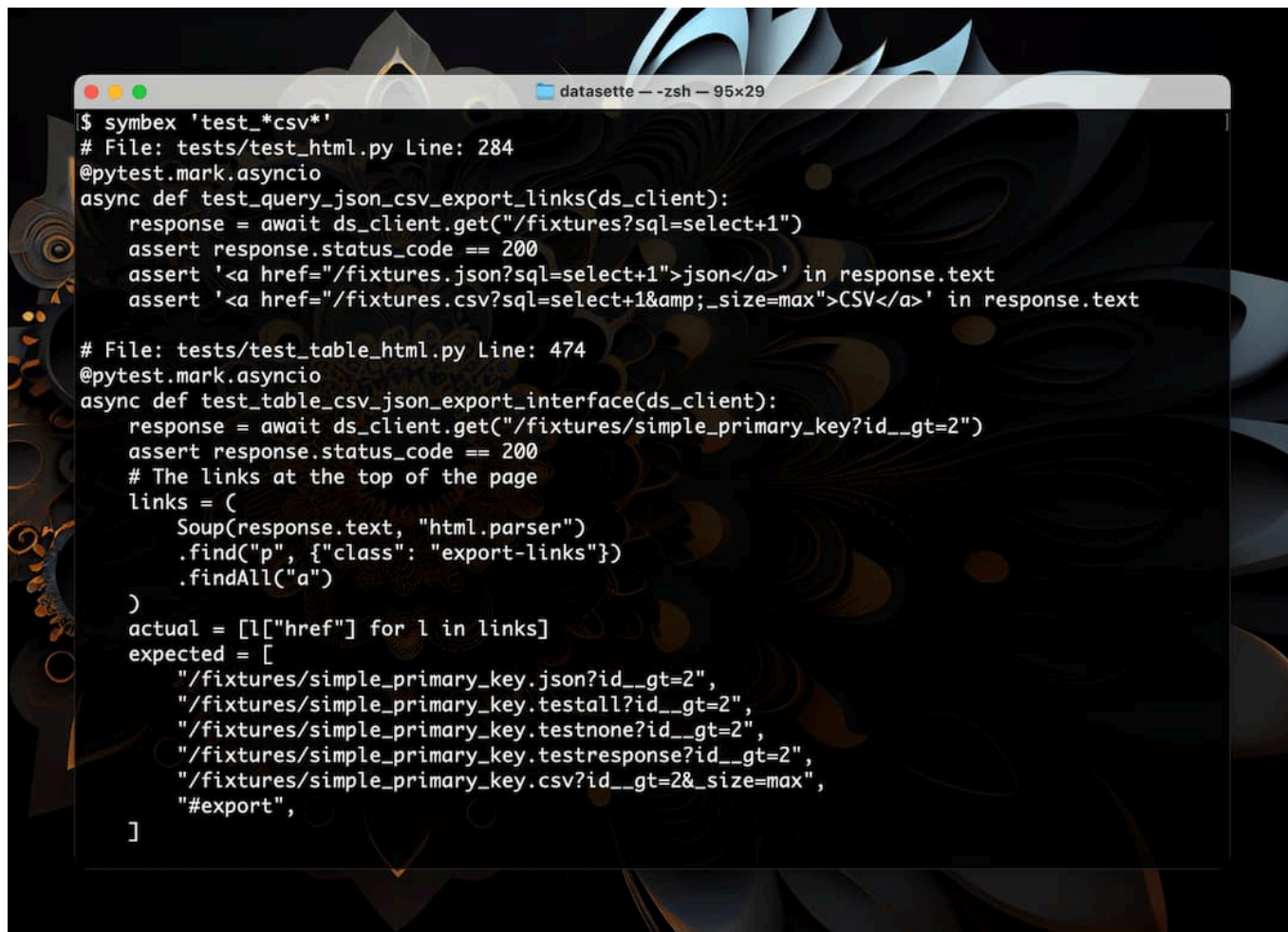
[LLM 0.4](#). I released a major update to my LLM CLI tool today—version 0.4, which adds conversation mode and prompt templates so you can store and re-use interesting prompts, plus a whole bunch of other large and small improvements.

I also released 0.4.1 with some minor fixes and the ability to install the tool using Homebrew: `brew install simonw/llm/llm`

[# 10:58 pm](#) / [cli](#), [projects](#), [releases](#), [ai](#), [openai](#), [generative-ai](#), [chatgpt](#), [llms](#)

June 18, 2023

Symbex: search Python code for functions and classes, then pipe them into a LLM



```
$ symbex 'test_*csv*'
# File: tests/test_html.py Line: 284
@pytest.mark.asyncio
async def test_query_json_csv_export_links(ds_client):
    response = await ds_client.get("/fixtures?sql=select+1")
    assert response.status_code == 200
    assert '<a href="/fixtures.json?sql=select+1">json</a>' in response.text
    assert '<a href="/fixtures.csv?sql=select+1&_size=max">CSV</a>' in response.text

# File: tests/test_table_html.py Line: 474
@pytest.mark.asyncio
async def test_table_csv_json_export_interface(ds_client):
    response = await ds_client.get("/fixtures/simple_primary_key?id__gt=2")
    assert response.status_code == 200
    # The links at the top of the page
    links = (
        Soup(response.text, "html.parser")
        .find("p", {"class": "export-links"})
        .findAll("a")
    )
    actual = [l["href"] for l in links]
    expected = [
        "/fixtures/simple_primary_key.json?id__gt=2",
        "/fixtures/simple_primary_key.testall?id__gt=2",
        "/fixtures/simple_primary_key.testnone?id__gt=2",
        "/fixtures/simple_primary_key.testresponse?id__gt=2",
        "/fixtures/simple_primary_key.csv?id__gt=2&_size=max",
        "#export",
    ]
```

I just released a new Python CLI tool called [Symbex](#). It's a search tool, loosely inspired by [ripgrep](#), which lets you search Python code for functions and classes by name or wildcard, then see just the source code of those matching entities.

[... [1,183 words](#)]

10:11 pm / [cli](#), [projects](#), [python](#), [ai](#), [generative-ai](#), [chatgpt](#), [llms](#), [symbex](#)

June 19, 2023

[Building Search DSLs with Django](#) (via) Neat tutorial by Dan Lamanna: how to build a GitHub-style search feature—supporting modifiers like “is:open author:danlamanna”—using PyParsing and the Django ORM.

8:30 am / [django](#), [dsl](#), [parsing](#), [python](#), [search](#)

June 22, 2023

Back then [in 2012], no one was thinking about AI. You just keep uploading your images [to Adobe Stock] and you get your residuals every month and life goes on — then all of a sudden, you find out that they trained their AI on your images and on everybody's images that they don't own. And they're calling it 'ethical' AI.

— [Eric Urquhart](#)

June 23, 2023

Every year, some generation of engineers have to learn the concepts of "there is no silver bullet", "use the right tech for the right problem", "you are not google", "rewriting a codebase every 2 years is not a good business decision", "things cost money".

— [sametmax](#)

[# 11:59 pm](#) / [software-engineering](#)

June 27, 2023

[Status of Python Versions](#) ([via](#)) Very clear and useful page showing the exact status of different Python versions. 3.7 reaches end of life today (no more security updates), while 3.11 will continue to be supported until October 2027.

[# 2:01 pm](#) / [python](#)

Weeknotes: symbex, LLM prompt templates, a bit of a break

I had a holiday to the UK for a family wedding anniversary and mostly took the time off... except for building **symbex**, which became one of those projects that kept on inspiring new features.

[... [1,120 words](#)]

[4:30 pm](#) / [projects](#), [ai](#), [weeknotes](#), [generative-ai](#), [llms](#), [symbex](#), [llm](#)

June 29, 2023

[abacaj/mpt-30B-inference](#). MPT-30B, released last week, is an extremely capable Apache 2 licensed open source language model. This repo shows how it can be run on a CPU, using the ctransformers Python library based on GGML. Following the instructions in the README got me a working MPT-30B model on my M2 MacBook Pro. The model is a 19GB download and it takes a few seconds to start spitting out tokens, but it works as advertised.

[# 3:27 am](#) / [open-source](#), [ai](#), [generative-ai](#), [local-llms](#), [llms](#), [llm-release](#)

June 30, 2023

[Databricks Signs Definitive Agreement to Acquire MosaicML, a Leading Generative AI Platform](#). MosaicML are the team behind MPT-7B and MPT-30B, two of the most impressive openly licensed LLMs. They just got acquired by Databricks for \$1.3 billion dollars.

[# 1:43 am](#) / [open-source](#), [ai](#), [generative-ai](#), [local-llms](#), [llms](#)

M	T	W	T	F	S	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		