September 2024

Search posts from September 2024 Search

82 posts: 10 entries, 49 links, 23 quotes

Sept. 1, 2024

uvtrick (via) This "fun party trick" by Vincent D. Warmerdam is absolutely brilliant and a little horrifying. The following code:

```
from uvtrick import Env

def uses_rich():
    from rich import print
    print("hi :vampire:")

Env("rich", python="3.12").run(uses_rich)
```

Executes that `uses_rich()` function in a fresh virtual environment managed by uv, running the specified Python version (3.12) and ensuring the rich package is available - even if it's not installed in the current environment.

It's taking advantage of the fact that `uv` is *so fast* that the overhead of getting this to work is low enough for it to be worth at least playing with the idea.

The real magic is in how `uvtrick` works. It's only 127 lines of code with some truly devious trickery going on.

That `Env.run()` method:

- Creates a temporary directory
- Pickles the `args` and `kwargs` and saves them to `pickled_inputs.pickle`
- Uses `inspect.getsource()` to retrieve the source code of the function passed to `run()`
- Writes *that* to a `pytemp.py` file, along with a generated `if __name__ == "__main__":` block that calls the function with the pickled inputs and saves its output to another pickle file called `tmp.pickle`

Having created the temporary Python file it executes the program using a command something like this:

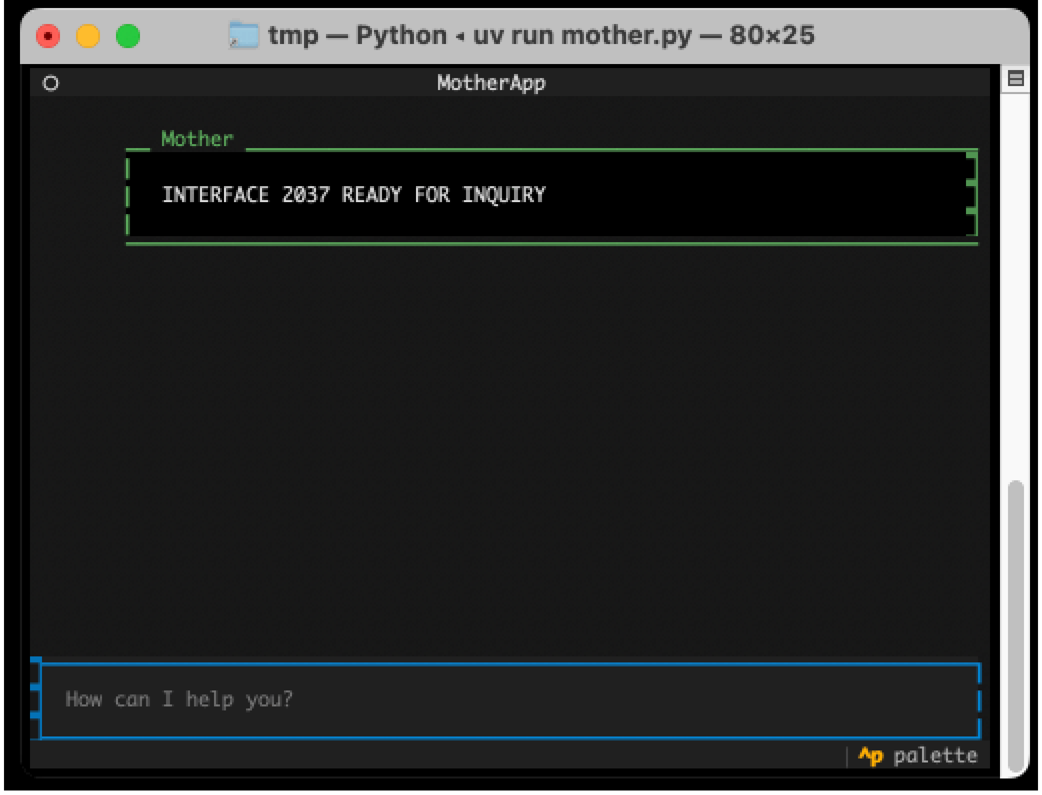
```
uv run --with rich --python 3.12 --quiet pytemp.py
```

It reads the output from `tmp.pickle` and returns it to the caller!

# 5:03 am / python, uv, vincent-d-warmerdam, rich

Sept. 2, 2024

Anatomy of a Textual User Interface. Will McGugan used Textual and my LLM Python library to build a delightful TUI for talking to a simulation of Mother, the AI from the Aliens movies:



```
INTERFACE 2037 READY FOR INQUIRY

How can I help you?
```

The entire implementation is just 77 lines of code. It includes PEP 723 inline dependency information:

```
# /// script
# requires-python = ">=3.12"
# dependencies = [
#     "llm",
#     "textual",
# ]
# ///
```

Which means you can run it in a dedicated environment with the correct dependencies installed using uv run like this:

```
wget
'https://gist.githubusercontent.com/willmcgugan/648a537c9d47dafa59cb8ece281d8c2c/raw/7aa575c389b31eb041ae7a909f2349a96ff
export OPENAI_API_KEY='sk-...'
uv run mother.py
```

I found the `send_prompt()` method particularly interesting. Textual uses `asyncio` for its event loop, but LLM currently only supports synchronous execution and can block for several seconds while retrieving a prompt.

Will used the Textual `@work(thread=True)` decorator, documented here, to run that operation in a thread:

```
@work(thread=True)
def send_prompt(self, prompt: str, response: Response) -> None:
    response_content = ""
    llm_response = self.model.prompt(prompt, system=SYSTEM)
    for chunk in llm_response:
        response_content += chunk
        self.call_from_thread(response.update, response_content)
```

Looping through the response like that and calling `self.call_from_thread(response.update, response_content)` with an accumulated string is all it takes to implement streaming responses in the Textual UI, and that `Response` object sublasses `textual.widgets.Markdown` so any Markdown is rendered using Rich.

# 4:39 pm / python, will-mcgugan, textual, llm, uv, rich

In Leak, Facebook Partner Brags About Listening to Your Phone's Microphone to Serve Ads for Stuff You Mention.

(I've repurposed some of my comments on Lobsters into this commentary on this article. See also I still don't think companies serve you ads based on spying through your microphone.)

Which is more likely?

1. All of the conspiracy theories are real! The industry managed to keep the evidence from us for decades, but finally a marketing agency of a local newspaper chain has blown the lid off the whole thing, in a bunch of blog posts and PDFs and on a podcast.
2. Everyone believed that their phone was listening to them even when it wasn't. The marketing agency of a local newspaper chain were the first group to be caught taking advantage of that widespread paranoia and use it to try and dupe people into spending money with them, despite the tech not actually working like that.

My money continues to be on number 2.

Here's their pitch deck. My "this is a scam" sense is vibrating like crazy reading it: CMG Pitch Deck on Voice-Data Advertising 'Active Listening'.

It does not read to me like the deck of a company that has actually shipped their own app that tracks audio and uses it for even the most basic version of ad targeting.

They give the game away on the last two slides:

> Prep work:
>
> 1. Create buyer personas by uploading past consumer data into the platform
> 2. Identify top performing keywords relative to your products and services by analyzing keyword data and past ad campaigns
> 3. Ensure tracking is set up via a tracking pixel placed on your site or landing page
>
> Now that preparation is done:
>
> 1. Active listening begins in your target geo and buyer behavior is detected across 470+ data sources […]
>
> Our technology analyzes over 1.9 trillion behaviors daily and collects opt-in customer behavior data from hundreds of popular websites that offer top display, video platforms, social applications, and mobile marketplaces that allow laser-focused media buying.
>
> Sources include: Google, LinkedIn, Facebook, Amazon and many more

That's not describing anything ground-breaking or different. That's how every targeting ad platform works: you upload a bunch of "past consumer data", identify top keywords and setup a tracking pixel.

I think **active listening** is the term that the team came up with for "something that sounds fancy but really just means the way ad targeting platforms work already". Then they got over-excited about the new metaphor and added that first couple of slides that talk about "voice data", without really understanding how the tech works or what kind of a shitstorm that could kick off when people who DID understand technology started paying attention to their marketing.

TechDirt's story Cox Media Group Brags It Spies On Users With Device Microphones To Sell Targeted Ads, But It's Not Clear They Actually Can included a quote with a clarification from Cox Media Group:

> CMG businesses do not listen to any conversations or have access to anything beyond a third-party aggregated, anonymized and fully encrypted data set that can be used for ad placement. We regret any confusion and we are committed to ensuring our marketing is clear and transparent.

Why I don't buy the argument that it's OK for people to believe this

I've seen variants of this argument before: phones do creepy things to target ads, but it's not exactly "listen through your microphone" - but there's no harm in people believing that if it helps them understand that there's creepy stuff going on generally.

I don't buy that. Privacy is important. People who are sufficiently engaged need to be able to understand exactly what's going on, so they can e.g. campaign for legislators to reign in the most egregious abuses.

I think it's harmful letting people continue to believe things about privacy that are not true, when we should instead be helping them understand the things that *are* true.

This discussion thread is full of technically minded, engaged people who still believe an inaccurate version of what their devices are doing. Those are the people that need to have an accurate understanding, because those are the people that can help explain it to others and can hopefully drive meaningful change.

This is such a damaging conspiracy theory.

1. It's causing some people to stop trusting their most important piece of personal technology: their phone.
2. We risk people ignoring REAL threats because they've already decided to tolerate made up ones.
3. If people believe this and see society doing nothing about it, that's horrible. That leads to a cynical "nothing can be fixed, I guess we will just let bad people get away with it" attitude. People need to believe that humanity can prevent this kind of abuse from happening.

The fact that nobody has successfully produced an experiment showing that this is happening is one of the main reasons I don't believe it to be happening.

It's like James Randi's One Million Dollar Paranormal Challenge - the very fact that nobody has been able to demonstrate it is enough for me not to believe in it.

# 11:56 pm / conspiracy, facebook, privacy, microphone-ads-conspiracy

---

Why I Still Use Python Virtual Environments in Docker (via) Hynek Schlawack argues for using virtual environments even when running Python applications in a Docker container. This argument was most convincing to me:

> I'm responsible for dozens of services, so I appreciate the *consistency* of knowing that everything I'm deploying is in `/app`, and if it's a Python application, I know it's a virtual environment, and if I run `/app/bin/python`, I get the virtual environment's Python with my application ready to be imported and run.

Also:

> It's good to use the same tools and primitives in development and in production.

Also worth a look: Hynek's guide to Production-ready Docker Containers with uv, an actively maintained guide that aims to reflect ongoing changes made to uv itself.

# 11:57 pm / packaging, python, virtualenv, docker, hynek-schlawack, uv

---

Sept. 3, 2024

Python Developers Survey 2023 Results (via) The seventh annual Python survey is out. Here are the things that caught my eye or that I found surprising:

25% of survey respondents had been programming in Python for less than a year, and 33% had less than a year of professional experience.

37% of Python developers reported contributing to open-source projects last year - a new question for the survey. This is delightfully high!

6% of users are still using Python 2. The survey notes:

> Almost half of Python 2 holdouts are under 21 years old and a third are students. Perhaps courses are still using Python 2?

In web frameworks, Flask and Django neck and neck at 33% each, but FastAPI is a close third at 29%! Starlette is at 6%, but that's an under-count because it's the basis for FastAPI.

The most popular library in "other framework and libraries" was BeautifulSoup with 31%, then Pillow 28%, then OpenCV-Python at 22% (wow!) and Pydantic at 22%. Tkinter had 17%. These numbers are all a surprise to me.

pytest scores 52% for unit testing, `unittest` from the standard library just 25%. I'm glad to see `pytest` so widely used, it's my favourite testing tool across any programming language.

The top cloud providers are AWS, then Google Cloud Platform, then Azure... but PythonAnywhere (11%) took fourth place just ahead of DigitalOcean (10%). And Alibaba Cloud is a new entrant in sixth place (after Heroku) with 4%. Heroku's ending of its free plan dropped them from 14% in 2021 to 7% now.

Linux and Windows equal at 55%, macOS is at 29%. This was one of many multiple-choice questions that could add up to more than 100%.

In databases, SQLite usage was trending down - 38% in 2021 to 34% for 2023, but still in second place behind PostgreSQL, stable at 43%.

The survey incorporates quotes from different Python experts responding to the numbers, it's worth reading through the whole thing.

# 2:47 am / open-source, postgresql, python, sqlite, surveys, pytest, psf, pydantic

---

> `history | tail -n 2000 | llm -s "Write aliases for my zshrc based on my terminal history. Only do this for most common features. Don't use any specific files or directories."`
>
> — **anjor**

# 3:01 pm / llm, llms, ai, generative-ai

Sept. 4, 2024

Qwen2-VL: To See the World More Clearly. Qwen is Alibaba Cloud's organization training LLMs. Their latest model is Qwen2-VL - a vision LLM - and it's getting some really positive buzz. Here's a r/LocalLLaMA thread about the model.

The original Qwen models were licensed under their custom Tongyi Qianwen license, but starting with Qwen2 on June 7th 2024 they switched to Apache 2.0, at least for their smaller models:

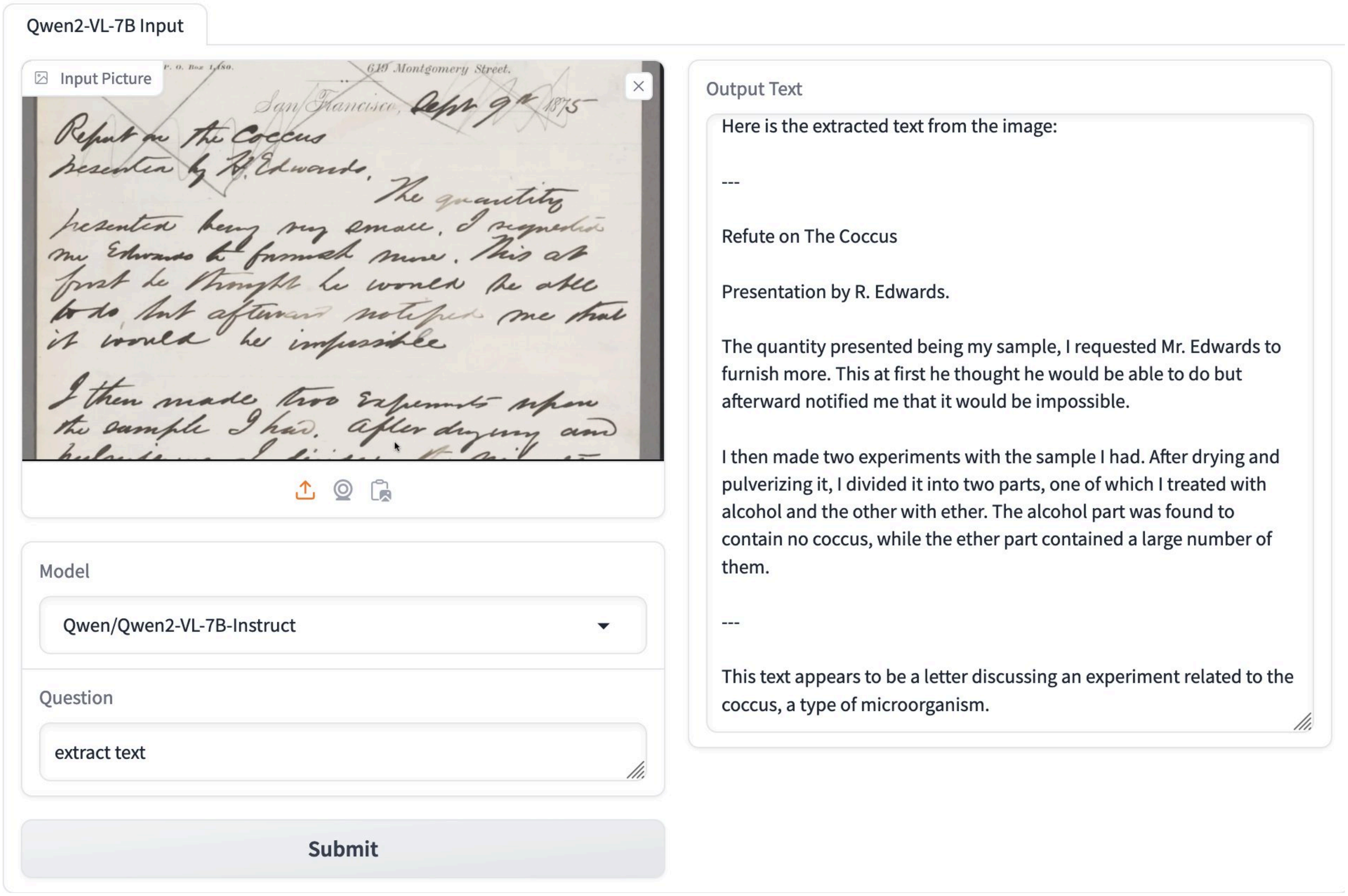
> While Qwen2-72B as well as its instruction-tuned models still uses the original Qianwen License, all other models, including Qwen2-0.5B, Qwen2-1.5B, Qwen2-7B, and Qwen2-57B-A14B, turn to adopt Apache 2.0

Here's where things get odd: shortly before I first published this post the Qwen GitHub organization, and their GitHub pages hosted blog, both disappeared and returned 404s pages. I asked on Twitter but nobody seems to know what's happened to them.

Update: *this was accidental and was resolved on 5th September.*

The Qwen Hugging Face page is still up - it's just the GitHub organization that has mysteriously vanished.

Inspired by Dylan Freedman I tried the model using GanymedeNil/Qwen2-VL-7B on Hugging Face Spaces, and found that it was exceptionally good at extracting text from unruly handwriting:



P. O. Box 1,450. 619 Montgomery Street.

San Francisco, Sept 9th 1875

Report on the Coccus
presented by H. Edwards.
The quantity
presented being very small, I requested
Mr Edwards to furnish more. This at
first he thought he would be able
to do but afterwards notified me that
it would be impossible

I then made two experiments upon
the sample I had. after drying and

Output Text

Here is the extracted text from the image:

Refute on The Coccus

Presentation by R. Edwards.

The quantity presented being my sample, I requested Mr. Edwards to furnish more. This at first he thought he would be able to do but afterward notified me that it would be impossible.

I then made two experiments with the sample I had. After drying and pulverizing it, I divided it into two parts, one of which I treated with alcohol and the other with ether. The alcohol part was found to contain no coccus, while the ether part contained a large number of them.

This text appears to be a letter discussing an experiment related to the coccus, a type of microorganism.

Model: Qwen/Qwen2-VL-7B-Instruct

Question: extract text

The model apparently runs great on NVIDIA GPUs, and *very slowly* using the MPS PyTorch backend on Apple Silicon. Qwen previously released MLX builds of their non-vision Qwen2 models, so hopefully there will be an Apple Silicon optimized MLX model for Qwen2-VL soon as well.

# 11:16 pm / ai, generative-ai, local-llms, llms, vision-llms, qwen, llm-release, ai-in-china

Sept. 5, 2024

OAuth from First Principles (via) Rare example of an OAuth explainer that breaks down *why* each of the steps are designed the way they are, by showing an illustrative example of how an attack against OAuth could work in absence of each measure.

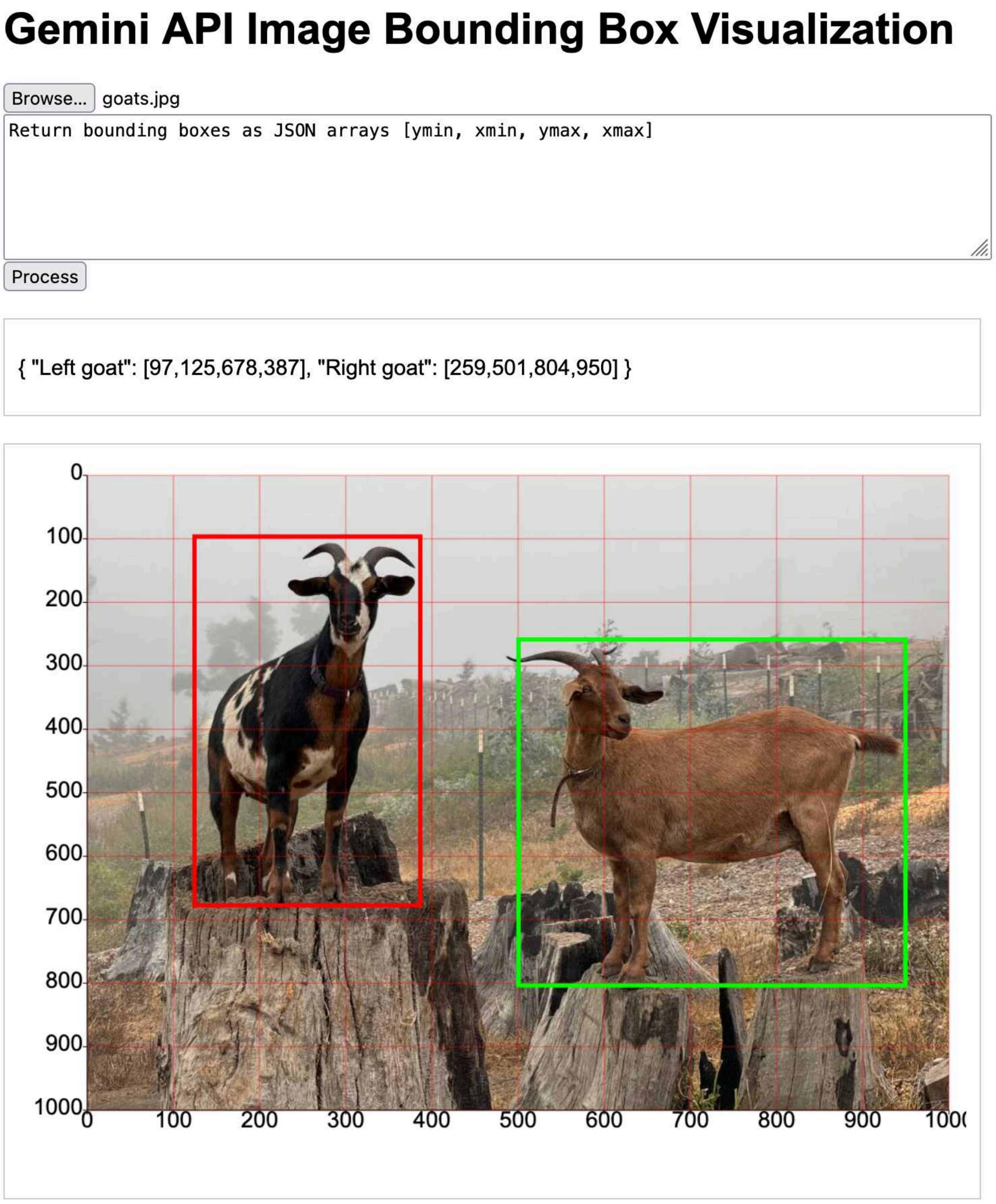
Ever wondered why OAuth returns you an authorization code which you then need to exchange for an access token, rather than returning the access token directly? It's for an added layer of protection against eavesdropping attacks:

> If Endframe eavesdrops the authorization code in real-time, they can exchange it for an access token very quickly, before Big Head's browser does. [...] Currently, anyone with the authorization code can exchange it for an access token. We need to ensure that only the person who initiated the request can do the exchange.

# 10:43 pm / oauth, security

Sept. 6, 2024

Calling LLMs from client-side JavaScript, converting PDFs to HTML + weeknotes



```
Return bounding boxes as JSON arrays [ymin, xmin, ymax, xmax]
```

```
{ "Left goat": [97,125,678,387], "Right goat": [259,501,804,950] }
```

I’ve been having a bunch of fun taking advantage of CORS-enabled LLM APIs to build client-side JavaScript applications that access LLMs directly. I also span up a new Datasette plugin for advanced permission management.

[... 2,050 words]

2:28 am / css, javascript, pdf, projects, ai, datasette, weeknotes, generative-ai, llms, anthropic, claude, gemini, claude-3-5-sonnet, cors

New improved commit messages for scrape-hacker-news-by-domain. My simonw/scrape-hacker-news-by-domain repo has a very specific purpose. Once an hour it scrapes the Hacker News /from?site=simonwillison.net page (and the equivalent for datasette.io) using my shot-scraper tool and stashes the parsed links, scores and comment counts in JSON files in that repo.

It does this mainly so I can subscribe to GitHub's Atom feed of the commit log - visit simonw/scrape-hacker-news-by-domain/commits/main and add `.atom` to the URL to get that.

NetNewsWire will inform me within about an hour if any of my content has made it to Hacker News, and the repo will track the score and comment count for me over time. I wrote more about how this works in Scraping web pages from the command line with shot-scraper back in March 2022.

Prior to the latest improvement, the commit messages themselves were pretty uninformative. The message had the date, and to actually see which Hacker News post it was referring to, I had to click through to the commit and look at the diff.

I built my csv-diff tool a while back to help address this problem: it can produce a slightly more human-readable version of a diff between two CSV or JSON files, ideally suited for including in a commit message attached to a git scraping repo like this one.

I got that working, but there was still room for improvement. I recently learned that any Hacker News thread has an undocumented URL at `/latest?id=x` which displays the most recently added comments at the top.

I wanted that in my commit messages, so I could quickly click a link to see the most recent comments on a thread.

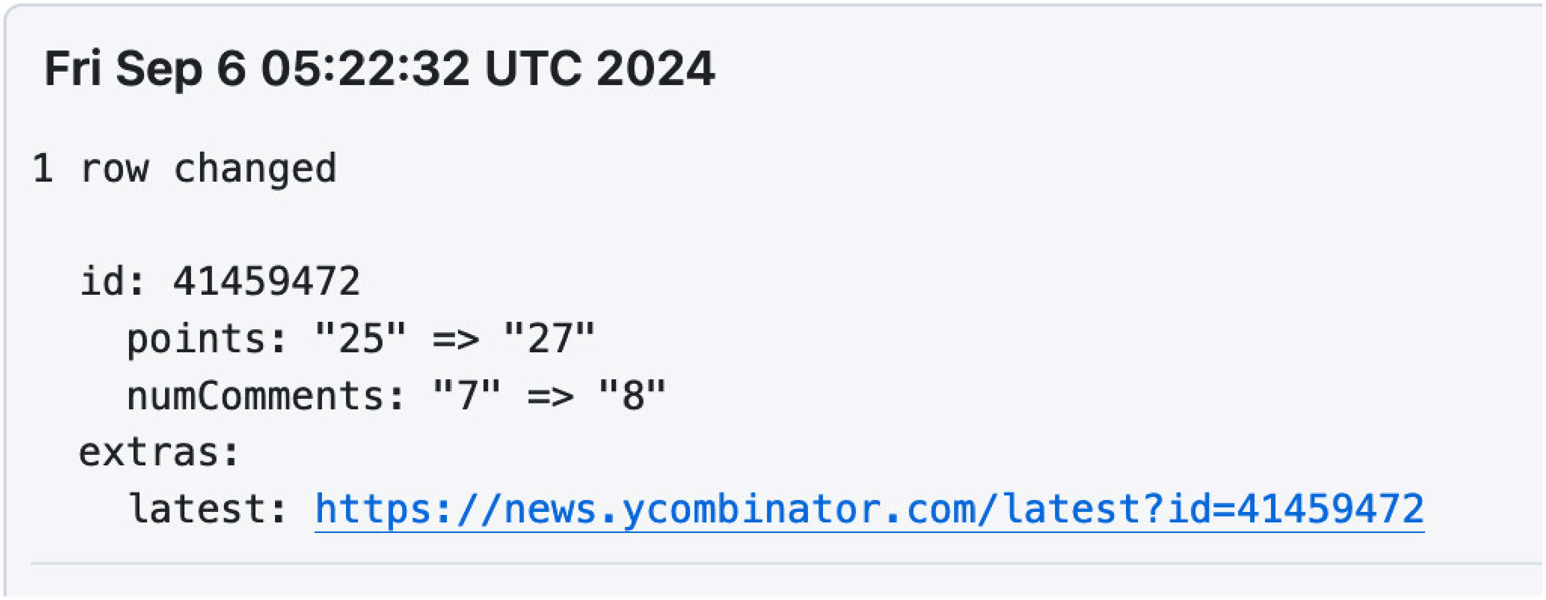
So... I added one more feature to `csv-diff`: a new --extra option lets you specify a Python format string to be used to add extra fields to the displayed difference.

My GitHub Actions workflow now runs this command:

```
csv-diff simonwillison-net.json simonwillison-net-new.json \
  --key id --format json \
  --extra latest 'https://news.ycombinator.com/latest?id={id}' \
  >> /tmp/commit.txt
```

This generates the diff between the two versions, using the `id` property in the JSON to tie records together. It adds a `latest` field linking to that URL.

The commits now look like this:



Fri Sep 6 05:22:32 UTC 2024

```
1 row changed

  id: 41459472
    points: "25" => "27"
    numComments: "7" => "8"
  extras:
    latest: https://news.ycombinator.com/latest?id=41459472
```

# 5:40 am / hacker-news, json, projects, github-actions, git-scraping, shot-scraper

Datasette 1.0a16. This latest release focuses mainly on performance, as discussed here in Optimizing Datasette a couple of weeks ago.

It also includes some minor CSS changes that could affect plugins, and hence need to be included before the final 1.0 release. Those are outlined in detail in issues #2415 and #2420.

# 5:55 am / projects, datasette

Docker images using uv's python (via) Michael Kennedy interviewed uv/Ruff lead Charlie Marsh on his Talk Python podcast, and was inspired to try uv with Talk Python's own infrastructure, a single 8 CPU server running 17 Docker containers (status page here).

The key line they're now using is this:

```
RUN uv venv --python 3.12.5 /venv
```

Which downloads the `uv` selected standalone Python binary for Python 3.12.5 and creates a virtual environment for it at `/venv` all in one go.

# 11:54 pm / python, docker, uv, charlie-marsh

Sept. 7, 2024

json-flatten, now with format documentation. `json-flatten` is a fun little Python library I put together a few years ago for converting JSON data into a flat key-value format, suitable for inclusion in an HTML form or query string. It lets you take a structure like this one:

```
{"foo": {"bar": [1, True, None]}
```

And convert it into key-value pairs like this:

```
foo.bar.[0]$int=1
foo.bar.[1]$bool=True
foo.bar.[2]$none=None
```

The `flatten(dictionary)` function function converts to that format, and `unflatten(dictionary)` converts back again.

I was considering the library for a project today and realized that the 0.3 README was a little thin - it showed how to use the library but didn't provide full details of the format it used.

On a hunch, I decided to see if files-to-prompt plus LLM plus Claude 3.5 Sonnet could write that documentation for me. I ran this command:

```
files-to-prompt *.py | llm -m claude-3.5-sonnet --system 'write detailed documentation in markdown describing the format used to represent JSON and nested JSON as key/value pairs, include a table as well'
```

That `*.py` picked up both `json_flatten.py` and `test_json_flatten.py` - I figured the test file had enough examples in that it should act as a good source of information for the documentation.

This worked really well! You can see the first draft it produced here.

It included before and after examples in the documentation. I didn't fully trust these to be accurate, so I gave it this follow-up prompt:

```
llm -c "Rewrite that document to use the Python cog library to generate the examples"
```

I'm a big fan of Cog for maintaining examples in READMEs that are generated by code. Cog has been around for a couple of decades now so it was a safe bet that Claude would know about it.

This almost worked - it produced valid Cog syntax like the following:

```
[[[cog
example = {
"fruits": ["apple", "banana", "cherry"]
}

cog.out("```json\n")
cog.out(str(example))
cog.out("\n```\n")
cog.out("Flattened:\n```\n")
for key, value in flatten(example).items():
    cog.out(f"{key}: {value}\n")
cog.out("```\n")
]]]
[[[end]]]
```

But that wasn't entirely right, because it forgot to include the Markdown comments that would hide the Cog syntax, which should have looked like this:

```
<!-- [[[cog -->
...
<!-- ]]] -->
...
<!-- [[[end]]] -->
```

I could have prompted it to correct itself, but at this point I decided to take over and edit the rest of the documentation by hand.

The end result was documentation that I'm really happy with, and that I probably wouldn't have bothered to write if Claude hadn't got me started.

# 5:43 am / json, projects, ai, generative-ai, llms, ai-assisted-programming, llm, anthropic, claude, claude-3-5-sonnet, files-to-prompt

Sept. 8, 2024

Teresa T is name of the whale in Pillar Point Harbor near Half Moon Bay

There is a young humpback whale in the harbor at Pillar Point, just north of Half Moon Bay, California right now. Their name is Teresa T and they were first spotted on Thursday afternoon.

[... 254 words]

1:04 am / photography, wildlife, half-moon-bay

---

uv under discussion on Mastodon. Jacob Kaplan-Moss kicked off this fascinating conversation about uv on Mastodon recently. It's worth reading the whole thing, which includes input from a whole range of influential Python community members such as Jeff Triplett, Glyph Lefkowitz, Russell Keith-Magee, Seth Michael Larson, Hynek Schlawack, James Bennett and others. (Mastodon is a pretty great place for keeping up with the Python community these days.)

The key theme of the conversation is that, while `uv` represents a huge set of potential improvements to the Python ecosystem, it comes with additional risks due its attachment to a VC-backed company - and its reliance on Rust rather than Python.

Here are a few comments that stood out to me.

Russell:

> As enthusiastic as I am about the direction uv is going, I *haven't* adopted them anywhere - because I want very much to understand Astral's intended business model before I hook my wagon to their tools. It's definitely not clear to me how they're going to stay liquid once the VC money runs out. They could get me onboard in a hot second if they published a "This is what we're planning to charge for" blog post.

Hynek:

> As much as I hate VC, [...] FOSS projects flame out all the time too. If Frost loses interest, there's no PDM anymore. Same for Ofek and Hatch(ling).
>
> I fully expect Astral to flame out and us having to fork/take over—it's the circle of FOSS. To me uv looks like a genius sting to trick VCs into paying to fix packaging. We'll be better off either way.

Glyph:

> Even in the best case, Rust is more expensive and difficult to maintain, not to mention "non-native" to the average customer here. [...] And the difficulty with VC money here is that it can burn out *all* the other projects in the ecosystem simultaneously, creating a risk of monoculture, where previously, I think we can say that "monoculture" was the *least* of Python's packaging concerns.

Hynek on Rust:

> I don't think y'all quite grok what uv makes so special due to your seniority. The speed is really cool, but the reason Rust is elemental is that it's one compiled blob that can be used to bootstrap and maintain a Python development. A blob that will never break because someone upgraded Homebrew, ran pip install or any other creative way people found to fuck up their installations. Python has shown to be a terrible tech to maintain Python.

Christopher Neugebauer:

> Just dropping in here to say that corporate capture of the Python ecosystem is the #1 keeps-me-up-at-night subject in my community work, so I watch Astral with interest, even if I'm not yet too worried.

I'm reminded of this note from Armin Ronacher, who created Rye and later donated it to uv maintainers Astral:

> However having seen the code and what uv is doing, even in the worst possible future this is a very forkable and maintainable thing. I believe that even in case Astral shuts down or were to do something incredibly dodgy licensing wise, the community would be better off than before uv existed.

I'm currently inclined to agree with Armin and Hynek: while the risk of corporate capture for a crucial aspect of the Python packaging and onboarding ecosystem is a legitimate concern, the amount of progress that has been made here in a relatively short time combined with the open license and quality of the underlying code keeps me optimistic that `uv` will be a net positive for Python overall.

Update: `uv` creator Charlie Marsh joined the conversation:

> I don't want to charge people money to use our tools, and I don't want to create an incentive structure whereby our open source offerings are competing with any commercial offerings (which is what you see with a lost of hosted-open-source-SaaS business models).
>
> What I want to do is build software that vertically integrates with our open source tools, and sell that software to companies that are already using Ruff, uv, etc. Alternatives to things that companies already pay for today.
>
> An example of what this might look like (we may not do this, but it's helpful to have a concrete example of the strategy) would be something like an enterprise-focused private package registry. A lot of big companies use uv. We spend time talking to them. They all spend money on private package registries, and have issues with them. We could build a private registry that integrates well with uv, and sell it to those companies. [...]
>
> But the core of what I want to do is this: build great tools, hopefully people like them, hopefully they grow, hopefully companies adopt them; then sell software to those companies that represents the natural next thing they need when building with Python. Hopefully we can build something better than the alternatives by playing well with our OSS, and hopefully we are the natural choice if they're already using our OSS.

# 4:23 pm / armin-ronacher, jacob-kaplan-moss, open-source, packaging, python, russell-keith-magee, rust, hynek-schlawack, mastodon, glyph, uv, astral, charlie-marsh, seth-michael-larson

---

Sept. 9, 2024

files-to-prompt 0.3. New version of my `files-to-prompt` CLI tool for turning a bunch of files into a prompt suitable for piping to an LLM, described here previously.

It now has a `-c/--cxml` flag for outputting the files in Claude XML-ish notation (XML-ish because it's not actually valid XML) using the format Anthropic describe as recommended for long context:

```
files-to-prompt llm-*/README.md --cxml | llm -m claude-3.5-sonnet \
  --system 'return an HTML page about these plugins with usage examples' \
  > /tmp/fancy.html
```

Here's what that gave me.

The format itself looks something like this:

```
<documents>
<document index="1">
<source>llm-anyscale-endpoints/README.md</source>
<document_content>
# llm-anyscale-endpoints
...
</document_content>
</document>
</documents>
```

# 5:57 am / cli, projects, tools, ai, prompt-engineering, generative-ai, llms, anthropic, claude, files-to-prompt

---

Why GitHub Actually Won (via) GitHub co-founder Scott Chacon shares some thoughts on how GitHub won the open source code hosting market. Shortened to two words: timing, and taste.

There are some interesting numbers in here. I hadn't realized that when GitHub launched in 2008 the term "open source" had only been coined ten years earlier, in 1998. This paper by Dirk Riehle estimates there were 18,000 open source projects in 2008 - Scott points out that today there are over 280 million public repositories on GitHub alone.

Scott's conclusion:

> We were there when a new paradigm was being born and we approached the problem of helping people embrace that new paradigm with a developer experience centric approach that nobody else had the capacity for or interest in.

# 5:16 pm / git, github, open-source

Sept. 10, 2024

> Telling the AI to "make it better" after getting a result is just a folk method of getting an LLM to do Chain of Thought, which is why it works so well.
>
> — **Ethan Mollick**

# 3:12 pm / prompt-engineering, ethan-mollick, generative-ai, ai, llms

---

Notes from my appearance on the Software Misadventures Podcast

I was a guest on Ronak Nathani and Guang Yang's Software Misadventures Podcast, which interviews seasoned software engineers about their careers so far and their misadventures along the way. Here's the episode: LLMs are like your weird, over-confident intern | Simon Willison (Datasette).

[... 1,740 words]

10:48 pm / blogging, podcasts, ai, prompt-engineering, generative-ai, llms, ai-assisted-programming, podcast-appearances

Sept. 11, 2024

Pixtral 12B. Mistral finally have a multi-modal (image + text) vision LLM!

I linked to their tweet, but there's not much to see there - in now classic Mistral style they released the new model with an otherwise unlabeled link to a torrent download. A more useful link is mistral-community/pixtral-12b-240910 on Hugging Face, a 25GB "Unofficial Mistral Community" copy of the weights.

Pixtral was announced at Mistral's AI Summit event in San Francisco today. It has 128,000 token context, is Apache 2.0 licensed and handles 1024x1024 pixel images. They claim it's particularly good for OCR and information extraction. It's not available on their La Platforme hosted API yet, but that's coming soon.

A few more details can be found in the release notes for mistral-common 1.4.0. That's their open source library of code for working with the models - it doesn't actually run inference, but it includes the all-important tokenizer, which now includes three new special tokens: `[IMG]`, `[IMG_BREAK]` and `[IMG_END]`.

# 10:18 pm / ai, generative-ai, local-llms, llms, mistral, vision-llms, llm-release

Sept. 12, 2024

Notes on OpenAI's new o1 chain-of-thought models

OpenAI released two major new preview models today: `o1-preview` and `o1-mini` (that mini one is not a preview)—previously rumored as having the codename "strawberry". There's a lot to understand about these models—they're not as simple as the next step up from GPT-4o, instead introducing some major trade-offs in terms of cost and performance in exchange for improved "reasoning" capabilities.

[... 1,568 words]

10:36 pm / ai, openai, prompt-engineering, generative-ai, llms, o1, llm-reasoning, llm-release

LLM 0.16. New release of LLM adding support for the `o1-preview` and `o1-mini` OpenAI models that were released today.

# 11:20 pm / projects, ai, openai, generative-ai, llms, llm, o1

> o1-mini is the most surprising research result I've seen in the past year
>
> Obviously I cannot spill the secret, but a small model getting >60% on AIME math competition is so good that it's hard to believe

— **Jason Wei**, OpenAI

# 11:45 pm / o1, generative-ai, openai, ai, llms, llm-reasoning

Sept. 13, 2024

> There is superstition about creativity, and for that matter, about thinking in every sense, and it's part of the history of the field of artificial intelligence that every time somebody figured out how to make a computer do something - play good checkers, solve simple but relatively informal problems - there was a chorus of critics to say, but that's not thinking.

— **Pamela McCorduck**, in 1979

# 7:49 am / ai, ai-history

> Believe it or not, the name Strawberry does not come from the "How many r's are in strawberry" meme. We just chose a random word. As far as we know it was a complete coincidence.

— **Noam Brown**, OpenAI

# 11:35 am / o1, generative-ai, openai, ai, llms

Sept. 14, 2024

Notes on running Go in the browser with WebAssembly (via) Neat, concise tutorial by Eli Bendersky on compiling Go applications that can then be loaded into a browser using WebAssembly and integrated with JavaScript. Go functions can be exported to JavaScript like this:

```
js.Global().Set("calcHarmonic", jsCalcHarmonic)
```

And Go code can even access the DOM using a pattern like this:

```
doc := js.Global().Get("document")
inputElement := doc.Call("getElementById", "timeInput")
input := inputElement.Get("value")
```

Bundling the WASM Go runtime involves a 2.5MB file load, but there’s also a TinyGo alternative which reduces that size to a fourth.

# 5:10 pm / go, javascript, webassembly

> It's a bit sad and confusing that LLMs ("Large Language Models") have little to do with language; It's just historical. They are highly general purpose technology for statistical modeling of token streams. A better name would be Autoregressive Transformers or something.
>
> They don't care if the tokens happen to represent little text chunks. It could just as well be little image patches, audio chunks, action choices, molecules, or whatever. If you can reduce your problem to that of modeling token streams (for any arbitrary vocabulary of some set of discrete tokens), you can "throw an LLM at it".

— **Andrej Karpathy**

# 7:50 pm / andrej-karpathy, llms, ai, generative-ai

Sept. 15, 2024

> [… OpenAI’s o1] could work its way to a correct (and well-written) solution *if* provided a lot of hints and prodding, but did not generate the key conceptual ideas on its own, and did make some non-trivial mistakes. The experience seemed roughly on par with trying to advise a mediocre, but not completely incompetent, graduate student. However, this was an improvement over previous models, whose capability was closer to an actually incompetent graduate student.

— **Terrence Tao**

# 12:04 am / o1, generative-ai, openai, mathematics, ai, llms

---

Speed matters (via) Jamie Brandon in 2021, talking about the importance of optimizing for the speed at which you can work as a developer:

> Being 10x faster also changes the kinds of projects that are worth doing.
>
> Last year I spent something like 100 hours writing a text editor. […] If I was 10x slower it would have been 20-50 weeks. Suddenly that doesn't seem like such a good deal any more - what a waste of a year!

It’s not just about speed of writing code:

> When I think about speed I think about the whole process - researching, planning, designing, arguing, coding, testing, debugging, documenting etc.
>
> Often when I try to convince someone to get faster at one of those steps, they'll argue that the others are more important so it's not worthwhile trying to be faster. Eg choosing the right idea is more important than coding the wrong idea really quickly.

> But that's totally conditional on the speed of everything else! If you could code 10x as fast then you could try out 10 different ideas in the time it would previously have taken to try out 1 idea. Or you could just try out 1 idea, but have 90% of your previous coding time available as extra idea time.

Jamie’s model here helps explain the effect I described in AI-enhanced development makes me more ambitious with my projects. Prompting an LLM to write portions of my code for me gives me that 5-10x boost in the time I spend typing code into a computer, which has a big effect on my ambitions despite being only about 10% of the activities I perform relevant to building software.

I also increasingly lean on LLMs as assistants in the research phase - exploring library options, building experimental prototypes - and for activities like writing tests and even a little bit of documentation.

# 8:58 am / ai, generative-ai, llms, ai-assisted-programming

---

How to succeed in MrBeast production (leaked PDF). Whether or not you enjoy MrBeast’s format of YouTube videos (here’s a 2022 Rolling Stone profile if you’re unfamiliar), this leaked onboarding document for new members of his production company is a compelling read.

It’s a snapshot of what it takes to run a massive scale viral YouTube operation in the 2020s, as well as a detailed description of a very specific company culture evolved to fulfill that mission.

It starts in the most on-brand MrBeast way possible:

> I genuinely believe if you attently read and understand the knowledge here you will be much better set up for success. So, if you read this book and pass a quiz I’ll give you $1,000.

Everything is focused very specifically on YouTube as a format:

> Your goal here is to make the best YOUTUBE videos possible. That’s the number one goal of this production company. It’s not to make the best produced videos. Not to make the funniest videos. Not to make the best looking videos. Not the highest quality videos.. It’s to make the best YOUTUBE videos possible.

The MrBeast definition of A, B and C-team players is one I haven’t heard before:

> A-Players are obsessive, learn from mistakes, coachable, intelligent, don’t make excuses, believe in Youtube, see the value of this company, and are the best in the goddamn world at their job. B-Players are new people that need to be trained into A-Players, and C-Players are just average employees. […] They arn’t obsessive and learning. C-Players are poisonous and should be transitioned to a different company IMMEDIATELY. (It’s okay we give everyone severance, they’ll be fine).

The key characteristic outlined here, if you read between the hustle-culture lines, is learning. Employees who constantly learn are valued. Employees who don’t are not.

There’s a lot of stuff in there about YouTube virality, starting with the Click Thru Rate (CTR) for the all-important video thumbnails:

> This is what dictates what we do for videos. “I Spent 50 Hours In My Front Yard” is lame and you wouldn’t click it. But you would hypothetically click “I Spent 50 Hours In Ketchup”. Both are relatively similar in time/effort but the ketchup one is easily 100x more viral. An image of someone sitting in ketchup in a bathtub is exponentially more interesting than someone sitting in their front yard.

The creative process for every video they produce starts with the title and thumbnail. These set the expectations for the viewer, and everything that follows needs to be defined with those in mind. If a viewer feels their expectations are not being matched, they’ll click away - driving down the crucial Average View Duration that informs how much the video is promoted by YouTube’s all-important mystical algorithms.

MrBeast videos have a strictly defined formula, outlined in detail on pages 6-10.

The first minute captures the viewer's attention and demonstrates that their expectations from the thumbnail will be met. Losing 21 million viewers in the first minute after 60 million initial clicks is considered a reasonably good result! Minutes 1-3, 3-6 and 6-end all have their own clearly defined responsibilities as well.

Ideally, a video will feature something they call the "wow factor":

> An example of the "wow factor" would be our 100 days in the circle video. We offered someone $500,000 if they could live in a circle in a field for 100 days (video) and instead of starting with his house in the circle that he would live in, we bring it in on a crane 30 seconds into the video. Why? Because who the fuck else on Youtube can do that lol.

Chapter 2 (pages 10-24) is about creating content. This is crammed with insights into what it takes to produce surprising, spectacular and very expensive content for YouTube.

A lot of this is about coordination and intense management of your dependencies:

> I want you to look them in the eyes and tell them they are the bottleneck and take it a step further and explain why they are the bottleneck so you both are on the same page. "Tyler, you are my bottleneck. I have 45 days to make this video happen and I can not begin to work on it until I know what the contents of the video is. I need you to confirm you understand this is important and we need to set a date on when the creative will be done." […] Every single day you must check in on Tyler and make sure he is still on track to hit the target date.

It also introduces the concept of "critical components":

> Critical components are the things that are essential to your video. If I want to put 100 people on an island and give it away to one of them, then securing an island is a critical component. It doesn't matter how well planned the challenges on the island are, how good the weather is, etc. Without that island there is no video.
>
> […]
>
> Critical Components can come from literally anywhere and once something you're working on is labeled as such, you treat it like your baby. WITHOUT WHAT YOU'RE WORKING ON WE DO NOT HAVE A VIDEO! Protect it at all costs, check in on it 10x a day, obsess over it, make a backup, if it requires shipping pay someone to pick it up and drive it, don't trust standard shipping, and speak up the second anything goes wrong. The literal second. Never coin flip a Critical Component (that means you're coinfliping the video aka a million plus dollars)

There's a bunch of stuff about communication, with a strong bias towards "higher forms of communication": in-person beats a phone call beats a text message beats an email.

Unsurprisingly for this organization, video is a highly valued tool for documenting work:

> Which is more important, that one person has a good mental grip of something or that their entire team of 10 people have a good mental grip on something? Obviously the team. And the easiest way to bring your team up to the same page is to freaken video everything and store it where they can constantly reference it. A lot of problems can be solved if we just video sets and ask for videos when ordering things.

I enjoyed this note:

> Since we are on the topic of communication, written communication also does not constitute communication unless they confirm they read it.

And this bit about the value of consultants:

> Consultants are literally cheat codes. Need to make the world's largest slice of cake? Start off by calling the person who made the previous world's largest slice of cake lol. He's already done countless tests and can save you weeks worth of

> work. […] In every single freakin task assigned to you, always always always ask yourself first if you can find a consultant to help you.

Here’s a darker note from the section “Random things you should know”:

> Do not leave consteatants waiting in the sun (ideally waiting in general) for more than 3 hours. Squid game it cost us $500,000 and boys vs girls it got a lot of people out. Ask James to know more

And to finish, this note on budgeting:

> I want money spent to be shown on camera ideally. If you’re spending over $10,000 on something and it won’t be shown on camera, seriously think about it.

I’m always interested in finding management advice from unexpected sources. For example, I love The Eleven Laws of Showrunning as a case study in managing and successfully delegating for a large, creative project.

I don’t think this MrBeast document has as many lessons directly relevant to my own work, but as an honest peek under the hood of a weirdly shaped and absurdly ambitious enterprise it’s legitimately fascinating.

# 2:37 pm / youtube, management, showrunning, leadership

page 1 / 3 next » last »»

2024 » September

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						