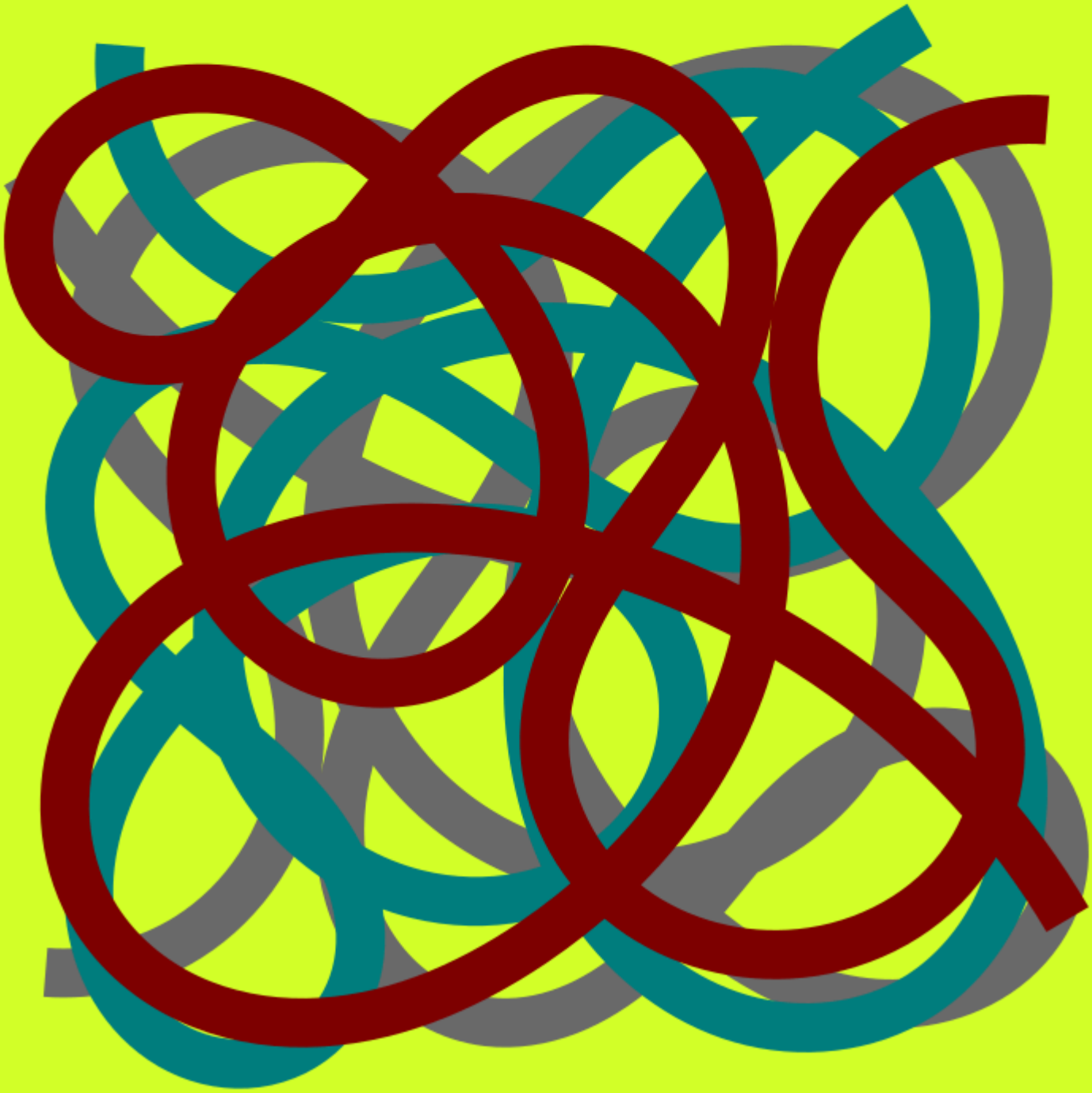
Diary 2025

Abdur-Rahmaan Janhangeer



Diary 2025

by Abdur-Rahmaan Janhangeer

build: 0.1.0

21. RAG paper review: Dense Passage Retrieval for Open-Domain Question Answering (3/50)
22. Achitecting AI Software systems book review
23. Vibecoding is not erasing open source, at all
24. Privacy: France shoots itself in the foot!
25. My first Nuitka PR
26. Mauritian incubators

1. Learn Python Programming Book Review

So, Packt sent me this book to give my opinion on it.

As always, Packt deserves some praise for consistently improve the quality of the books.

Let's review Learn Python Programming by Fabrizio Romano and Henrich Kruger

I like the book for it's warm introduction to programming, Python uses in the real-world and the classic installation hand-holding.

It's off to a great start by using Python3.12, though many packages haven't caught up to it. Even Google Gemini package (google-generativeai) supports only upto Python3.11, but, i believe it a matter of time. It also states that Python can be run as a service, which sparks a lot of ideas.

The first part is hands-on practice on the Repl, brilliant idea. Books tend to focus on code in files. But, beginners enjoy being able to do something with instant feedback and focusing on the language rather than being interrupted by file runs.

The reasons for using functions are compelling, succinct and relatable. Great addition of good

practices. I like the way it tackles uncanny topics like generators. Kudos for talking about TOML!

I was wondering why it included cryptography in this book, then i saw the section about JWT, nice shot! It’s essential that developers learn about JWT in a non-intimidating way.

The book is a nice starter toolkit, talks about relevant concepts, adding libraries where needed like pytest for tests, and typing. I also laud the inclusion of SQLAlchemy and a section for data science.

This book approaches fundamentals over frameworks, which is great. It refuses to teach Flask or Django (but is more Django prone and uses Django as example, just saying), but teaches requests, data interchange, formats and databases. Good move.

The book is good and thoughtful. It’s a recommended buy if you want to ramp up a solid foundation in Python quickly (Not sponsored comment).

This book, as all books are long and i recommend speeding up on the easy parts.

Link to book in comment.

2. What happens when your ads data gets leaked

This picture shows what happens when your ads data gets leaked. If we tie them to you, we know exactly where you went and at what hotel you stayed, for example. As the dots tell of this unfortunate person.

This is a privacy issue of course. But, in the Gravy Analytics leak, we witness first-hand the consequence of using tech giants services instead of privacy-focused solutions.

The sample the hacker leaked included millions of location data points. Areas covered include military bases, the White House, the Vatican etc. You are not safe just because you are in a physically restricted area.

The data was sourced from some apps including:

com . facebook . katana (Facebook) com . instagram . android (Instagram) com . whatsapp (WhatsApp) com . snapchat . android (Snapchat) com . tiktok . android (TikTok) com . google . android . youtube (YouTube) com . spotify . music (Spotify) com . king . candycrushsaga (Candy Crush Saga) com . miniclip . eightballpool (8 Ball Pool) com . supercell . clashofclans (Clash of Clans) com . supercell . brawlstars (Brawl Stars) com . imangi .

templerun2 (Temple Run 2) com . ea . gp . fifamobile (FIFA Mobile) com . epicgames . fortnite (Fortnite) com . mojang . minecraftpe (Minecraft) com . zynga . FarmVille3 (FarmVille3) com . netflix . mediaclient (Netflix) com . tinder (Tinder) com . microsoft . office . outlook (Outlook) com . discord (Discord)

It’s a hard jolt to know that our favourite apps can be incredibly dangerous if conditions meet.

For example, if you have been in your house, we can tie the location to you and follow you all along. It has been done with the leak.

Connecting random points is just as fruitful. Let’s say you are looking for someone who visited a supermarket, a phone store and a random house. You only find a weak link like who visited the house. Apply this to points who were tracked inside the white house and we know who they were and of course where they roam.

I liked the way Baptiste Robert analysed the data points, link in comment.

3. Battling Reddit Toxicity and Winning Big

Last month i battled Reddit toxicity and won BIG.

I was posting an article as usual, when the first comments started rolling in.

“Random schmuck advertising their substack. One comment in 10 minutes, but six upvotes. Hmmmmmmmmmmmmm”

I was deeply hurt being called a `random schmuck` by a random someone. And the post was getting downvoted at an alarming rate. Someone even started a debate on the legitimacy of self-promotion with proponents on both sides arguing. I had to cook something up.

I politely explained how i was not a random someone and illustrated how i help the Python community IN REAL LIFE.

This turned the tide, burying the comment, with up-votes returning again.

They were still some salty comments like “Yeah so they do a lot of pretty standard stuff, in other words” and “Could’ve just written “be a professional”“, but by that time the post was doing great and i was tackling comments with thoughtful replies.

Sometimes some comments of mine did get down-voted for no reason at all, but i did not care.

People did not mind it was me ‘advertising’ my post as the content seemed to be good.

The post got me the Top 1% poster badge on a top 1% subreddit.

So, yes, content is king, even on Reddit.

4. Making articles go viral

How to make articles go viral? It depends on the goal you want to achieve.

By now i've got few Hackernews front page hits and countless selections in newsletters.

Platform plays a great role. The CIA article i shared got feeble interest on LinkedIn and Twitter, but got popular on r/Python

The SQLite book performed well pretty much everywhere. Someone else posted it on Hackernews. On Twitter it took off when a db expert posted about it.

It is interesting to note that there is a weak link between virality and good content. Virality means it is read widely. But, if you post good content that most people don't understand, it won't go viral. So, the balance must be struck between content that many people can relate to and depth.

As software engineer i bet on good content as, by writing them you grow at the same time. I reserved my newsletter (Luminotes) for technical deep dives. Not many people are interested in it. But, people who are in the field do see the potential, wondering why it's not read more widely. I got a recommendation from a Google eng (Michal Pitr) and exclusive permission from a university professor (Murat Dogruel) to publish his

techniques to the wider public for the first time. I would not have received those without good content.

As i like Python and Python is popular, some deep dives do go viral. Like the Ruff internals article.

As blogger, i don't have goals to tick. I got all bloggers want to achieve.

But, i keep going as i was writing those articles for myself all that time. Sure i tuned some for virality, but it was content for my own consumption first.

So i guess, writing deep dives for oneself is a good bet that outlives the hackneyed goal of blogging. If your article does not take off, don't worry. Maybe the title was not well-crafted. Or you choose the wrong platform. Or most people won't find it useful.

But, keep going.

5. OpenAi is a felon company

OpenAI is a shockingly felon company, with pirate Altman at the Helm. These past days we've been seeing how brutally unethical OpenAI has been as a company and how Altman, despite great tech advises, is a despicable individual.

Getting data at all costs has been the company's motto since very early. Since 2013, piracymonitor . org has been warning [1] that OpenAi might have trained on copyrighted books from the Eye, as per a lawsuit. Eye's Books1 and Books2 datasets are almost 15 percent of what GPT-3 learned from [2].

OpenAI's scrapers are comical. Any beginner web-scraping programmer might bring websites down. But not people working for billion-dollar companies. OpenAI used 600 IPs to scrape data from a Ukrainian company [3]. Triplegangers CEO, Oleksandr Tomchuk labelled it as a DDOS attack.

ChatGPT-3.5 and ChatGPT-4.0 are accused of using HUNDRED of MILLIONS of personally identifiable information collected from people around the world WITHOUT their consent [4]. It's a continual practice it seems. Private info include private conversations, medical data and any data it could lay it's hands on. Stealing data while raising billions: double buccaneering?

It also mind-bogglingly spies on it's own users and sells their data [4]. Individual users of applications

and devices that have integrated ChatGPT-4 saw their chat logs, typed searches, key strokes, IP addresss, location and payment info being collected. If this was not bad enough, this info was also shared with others.

If this was not bad enough (second time), they also collected data when you interacted with social media sites …

One wonders then, if the captain of the ship is a nice person? A saint of our time driving a messianic mission? Sam Altman, it seems engaged in inappropriate behaviors with his sister. Ok the sister is crazy you say. But, recently, a whistleblower (Sudhir) was found dead and ruled as a suicide. Upon a second examination by the family, they laid grounds for a possible murder case. Sudhir's mother recalls his son pictured Altman as a liar [5], confirming maybe the time when ex-board members called Sam a liar and a psychological abuser (against employees) [6].

When faced with hurdles, OpenAi just removes them, such as firing Leopold Aschenbrenner for raising safety concerns [7]. Let's hope the angelic company did not hack Sudhir out of it's way.

This all surely calls to keep a two foot long stick with you to measure the distance between you and Sam Altman and to give it to the judge sending him to the gallows to thrash him.

6. Is it still worth it to write blog posts?

Is it still worth it to write blog posts in the age of LLMs? Especially since every info is available at the spin of a prompt?

I think yes, we still need to write. Because writing clarifies understanding and organizes ideas. A blog post is a share of the writing. Maybe the particular way it was woven helps some people understand the topic better. Writing builds the brain.

Writing is a record of particular recipes. How to do X? Maybe with LLMs we need to remember the prompt or retype and wait for an answer. A blog post is a cached, personilized response.

Writing broadens one's perspective. When wriring, in addition to knowing the answer, it prompts you to dig further and to consider more resources.

And, it improves language. You become better at articulating concepts, picking vocabulary, constructing analyses and communicating ideas. This directly helps beyond the screen, in real life.

Would you add more? Or have other ideas?

7. LLM caught not being able to count letters in a word

I caught ChatGPT not being able to count, a strawberry moment. It hallucinated phrases for me. Imagine me gobbling down that phrase for something critical, wondering why i used up my 3 attempts.

I gave it a simple case: I need to memorize 3 7 7 4 2 6 4 1 5 3 7 5.

Then i asked it to help me using letters. It gave me the obvious:

“You can create a phrase where the number of letters in each word matches the digits”

Of course, i am not going to pour an afternoon on the task. When it provided the first example, i noticed it was mistaken.

I tried again, and again, and again. And it messed up.

LLMs cannot count?

8. 327K pypi downloads

327, 995 downloads on Pypi for all my projects combined. After some 7 years of OpenSource, i'm happy to see some projects being used.

- 👉 88k: My signature project is shopyo, a framework for rapid Flask development. It tops the list. I talked at few conferences about it, including EuroPython.
- 👉 69k: Then comes Hooman, a wrapper around pygame for cleaner codebases, featured on r/pygame.
- 👉 58k: Meteomoris is an API for extracting content from our meteo's website. Mauritians are using it a lot, it seems!

Being useful helps a lot. Fastoo for example is a prototype of shopyo for FastAPI. I did not have time to complete it. Even then it got 11k.

If you want to get into OpenSource, just start. The time is now!

9. Calling yourself an engineer

When can you call yourself an engineer? How to eliminate the impostor syndrome definitely?

First of all, when we call someone an engineer, it has to be in a field. As one person cannot be an engineer across all fields as computing nowadays is too vast.

This not only applies to computing. We call Von Neumann the last genius in terms of mastery over several fields like Maths, Computing, Physics and Chemistry. What we call software engineering or computer science is a basic attempt at distilling the fundamentals of computing in 3 - 4 years.

A true engineer has to be in a field as industries operating in a field are normally touching advanced concepts in this particular field. And they need people who specialize in it to produce results and push boundaries. Sure they do take novices but don't expect them to be immediately useful.

To push boundaries and rear achievements, we must first understand what's going on at the lowest level possible. This clarity is what makes an engineer an engineer. Once this clarity settles in, problems are identified in shorter amount of times and shortcuts are identified.

If you feel the impostor syndrome, that's because yes, you are an impostor. Maybe it's not an issue but maybe it must be resolved by investing the appropriate amount of time to master the subfield you are operating in or to confront the task at hand.

Since codebases are huge and knowledge is vast, constantly learning deeply about what you come into contact is what makes you a top engineer. There is no incentive for deep focus as it oftentimes feels useless, but time and again you reap the rewards.

So, first yes, be curious. Second go deep. And third be constant about it. Then you will compound several deep area of expertise. There might not be an immediate monetary benefit, nor does your job mandate it but after sometime, you will see an awesome upgrade in your quality of life as an engineer.

10. Glauber Costa: A legend!

What i know about SQLite today and me writing the only free book about it's internals is a testimony of Glauber Costa's vision. He was bold enough to undertake the un-thought of: forking a software tested on billions of devices and running on trillions. He got the heat on the orange site for it, it did not age well of course and they were clueless tenrecs for once.

My criticism of SQLite COC lies in it's restricted nature. Though i don't agree with 50 and 63 "Love chastity.".

I always wanted to contribute to SQLite. And it was open they said. But to my dismay, it was closed with only the source available. A great label for this type of project is source open rather than open source. It's high time people stop promoting SQLite as an OpenSource project.

People are free to define the scope and type of their projects, but they should also bear the response to it, simple. LibSQL was a great contender and still is, but it lacked the community push behind. People just could not related to what was ongoing or maybe C is not so appealing? Turso has Rust as it's wand of choice.

Turso was bright enough to capitalize on Pekka's complete Rust rewrite as open source figurehead.

Forking is disdained but, it's in the spirit of OpenSource. Forking opens the doors to more contributors. One might argue that it splits the community and if all the contributors banded together in one project it would have been far better. True, except that everyone banding together reduces the amount of talents. Forking creates talents. On this i bet high on the future of SQLite.

My fear about Limbo by Pekka Enberg is stability. SQLite achieved world-class bug squashing by being deployed everywhere massively. Will Limbo be able to equate it ever? No, i don't think so. But, Limbo has a hidden card: DST (Deterministic Simulation Testing). Ensuring that simulations produce the same output for a given input, makes testing repeatable and verified. I am amazed by TigerBeetle's use of DST. This substitutes the need for crashing the software against a horde of users to ensure all is well and good.

But, maybe, secretly we are thankful to SQLite for keeping it closed and guarded by moral ashlars of good conduct. Without the gate-keeping we would probably not have the drive to bring about a revolution.

11. The most future-proof career move right now: be human

The most future-proof career move right now, is to be human. What a human can do with a computer, it can be observed, replicated and refined. If you rely on a computer skill to survive, sorry my friend, you can be replaced.

Even low-level engineering can be replicated. At the very least, embedded engineers can use Ai to ping-pong ideas and come up with inspirations.

Being low-level does not mean we won't use Ai or Ai won't make the task easier. A group of researchers taking interest in a field are expected to find decent solutions. Why Ai fails at specialized tasks is that LLM authors try to tackle the most common tasks. Focus refines niche tasks.

Critical fields are not insulated from tech advances, we just have to make sure there is a way to verify that the generated code is good enough. And it's not far, the time when this will be done.

What can't be replaced is you, the human. Writing code was always a surrogate for natural human expression. The clearer a human expresses himself, the clearer should the output be. We observe this with LLMs. The clearer the prompts, the clearer the results, mostly at least.

Humans will always rule the world. And humans like humans to interact with. The loftier a human is, the easier it is for him to interact with others. Tools are subservient to man. And they flow according to the influence of great men.

People, addicted to laziness and wanting to be done away with labour, will increasingly consider learning a ton of intricate engineering knowledge as useless. Let the Ai do it, it's horrible maybe, but it works, fine, we don't need to dive that much. Saying this from observing the current attitude of university professors to Ai. They tell students it's only important to know how it works, relying on the teachings of Ai.

Tech always come and go. Tech always fits human needs. It's humans who create culture. To be timeless and obsolete-proof, you just need to be a great human, useful in society, pleasant in company, with high, purposeful goals, austere in approach, with sound principles, constant advices and a relentless, altruistic, unbending will to better society.

The fruits of such an approach is that humans reward you in different ways, giving access to whatever resources they hold. But, adopting the described approach requires not relying on people's piggy banks. Job security, financial security comes as a by-product.

Do we aim to work or do we aim to make money? Do we aim to have money or do we aim to live pleasantly? Is a 9 to 5 living representative of the potential in a human being or is it a powerful sedative taking you on a ride till death?

12. Mistral: European Ingenuity?

Before you congratulate Mistral as an achievement of Europe's ingenuity and a solid response to the dazzling US ecosystem, please do yourself a favour and read this part:

"Founded in April 2023 by engineers formerly employed by Google DeepMind[4] and Meta Platforms,"

When Mistral fans blow the hype about `finally we beat the Americans`, people laugh them off. True, it's unfair to compare the US and Europe as the US is big, and Europe small. US is one country while Europe has many countries, making collaboration difficult. True. Let's say it's true.

But, when you see the innovation happening in the US, it is very concentrated. Leading Ai companies are surprise, surprise, located in San Francisco. The Silicon Valley itself is indeed very small. I'd say that the US has developed, since long, a great culture. And they are till now, reaping it's fruits.

The difference between the US and Europe i think is a matter of people and mentality. US people are open and very relatable. People from Europe 100% pretend to be open and relatable. Europe is rigid, the US is flexible. The US also i would argue, loves to attract immigrants. When you are at the edge, worldwide technical talents flock to you.

The US is culture-less and likes to discover and experience different cultures. Europe has a cultural legacy to protect and perpetuate.

I talked to people in SF. Even those from other countries who went and worked in Europe. Europe did not sit well with them. Even and specially France. Everything seems to be off in France. First of all the people. This is not criticism, this is reporting.

The people in the US (SV) are friendly. They like to talk, make friends and are very practical. Europe still has consideration but they pointed out to me that's nowadays it's not special at all. Be it places, be it culture, be it cuisine. Now people have a window to the world. There are far better cuisine than french cuisine for example.

Europe is doing something not so right, Europoor is even a term nowadays. Part of life is having fun. If work is boring, you don't want slow-moving people making it even less fun. You'd want to surround yourself with people who like to move forward, who are open to ideas and believe in solid engineering. You don't want to work with people who drag dragon legal frameworks with them to the office.

So, Mistral, bon vent and wish you more fund raisings, especially from US VCs.

13. Contributing to open source puts you miles ahead of the average developer

I brazenly believe that contributing to open source puts you miles ahead of the average developer.

Saying this after collaborating with developers for years. Contributing to open source nurtures some great habits and etches them in your brain's muscle memory. People used to contributing to open source have good developer reflexes as secondary nature.

The main thing about open source is that it trains you to become considerate. You care for collaborators, users as well as your future self. You modulate the project's direction towards stellar maintenance. You care about engineering a swift and painless onboarding experience.

Developers who don't contribute to open source miss basics such as writing a good README. A good README is the first step for open source projects as it is the project's window display. It boosts or dunks adoption.

A good README ensures that even when the developer is not around people can come and setup the project. It is a basic litmus test for your professionalism. I would be less inclined to work with someone who has to be begged hundred of times to craft a good README, don't you think?

Constantly contributing to open source makes you a generally good developer in the sense that you are exposed to production-grade software. This of course if you contribute to software that's used. You get an idea of the level of professionalism needed and learn about the importance of foresight. You are always on the lookout for things that could go wrong and how to handle them as gracefully as possible.

If you are exposed often, you also learn the very latest standard for your tech-stack or field. For example: what is the best way to run CI or the best test framework or better libraries. Oftentimes contributors propose great changes and you sit back and learn. Oftentimes you just spy on competitors and learn.

Often contributing to a quality project means that you routinely are in a mature contributing cycle. You are sometimes the reviewer, sometimes the one correcting your own PR, sometimes going back to the drawing board. You also write based documents explaining your idea or feature and why you think this should go forward.

Then you also have the ability to communicate asynchronously and drive initiatives forward. You don't have constant meetings. Sure the setting is different and the pace also but, you bet on being

able to communicate clearly and objectively, without office politics.

Open source also teaches about dealing with people. You need to deal with contributors, orient them for proposals and be willing to sacrifice part of what you wanted. You learn to codify your vision, hoping for at least the evolution looking like some 80% of what you intended. You also learn to take ownership and look how the overall picture impacts your part.

Dealing with open source people in commercial setups is really nice as you have someone who has a decent level, can communicate and collaborate smoothly with autonomy.

14. The Dodo coming back? Dire bollocks

We might see the Dodo soon, after the dire wolf come-back. And that too within a decade. The company Colossal Biosciences has the Dodo as 2nd or 3rd top priority. Here's how they intend to do it.

In case you are wondering, the Dodo comes from Mauritius, where i live.

Mauritian authorities expressed much enthusiasm at the thought of generating revenue through eco-tourism says Beth Shapiro, a noted ancient-DNA researcher who is now the chief science officer at Colossal, says, reminding the time when she visited Mauritius.

The person in charge of the Dodo project at Colossal, Anna Keyte, led a team at U.C. Santa Cruz that sequenced the Dodo genome. Interest in the Dodo has been around since sometimes it seems.

People got crazy over dire wolves as they are icons of the Game of Throne series, loved by many people. When people heard they were coming back, it made big headlines.

But, the science is based on deception. How the thing works is that it imitates dire wolves.

Elinor Karlsson, a program director at a joint Harvard and M.I.T. facility and an expert on wolf and dog genetics, asked Beth: "Why are you calling

this a dire wolf when it's a gray wolf with seventeen or eighteen changes in its DNA?".

Beth responded: "We've succeeded in creating the phenotype of a dire wolf." A phenotype is the set of observable characteristics of an individual resulting from the interaction of the it's genetic constitution with the environment. In programming we say if it quacks like a duck it's a duck, not true at all in this particular context.

Bringing back the Dodo is as deceptive. Keyte, the Dodo program director, has been looking at the ten thousand genetic differences between the Dodo and it's extant relatives to identify which gene is responsible for what characteristic. They also looked at the Solitaire, from Rodrigues now extinct and found that both are relatives of the pigeon.

They assume that pigeons came to Mauritius and Rodrigues and became larger and decided that they no longer needed to fly based on a flight-related gene that had become inactive in these two.

So, they are thinking of disactivating a flight-gene in the Nicobar pigeon to recreate the Dodo. Resurrecting the Dodo is thus pure sensationalism and is deceptive labelling at best.

I took the materials from "The Dire Wolf Is Back, The New Yorker"

Below is one of the best representations of the Dodo from Ustad Mansur, an Indian Painter who lived during the time of Jahangir. He actually saw the Dodo and is one of the best representations.

Current paintings base themselves on it. Even the color is believed to be very accurate.

15. Thanks to LLM, we must now slap a login page everywhere

I strongly advocate for slapping a login page everywhere on the internet to wall off bots. As companies driven by greed maliciously snub robots.txt, we have no choice but to wall off the internet.

People have always intended to help others when they put a recipe online or a fix to a problem. They could never fathom a day would come when their content would fatten up morally shipwrecked companies.

Companies should have thought of a way to compensate authors for copyrighted works and give credits at the very, very least. But, LLMs by design cannot currently attribute outputs to specific works. It's outrageous that in seconds they ingest and thieve years of human labor. And go on carrying business like nothing happened. They just don't care.

We find ourselves in a situation where we should starve LLMs until avaricious CEOs consider that right and wrong exists and we should choose the right path. This is the weaker option if we want to

send a strong signal. It would have been far better to boycott these companies upfront.

With nearly 1% of cloudflare traffic being bots, cloudflare advocates for intelligent bot trapping by spinning considerate honeypots. This is the way forward.

They steal from us, repackage our content and sell to us the same thing at some $200 per month. We should be very foolish to fall into the trap, pay and glorify these buccaneers. Their hands have become too red for them to backtrack. Celebrating a company's 50 years with a spotlight on looted artifact is deep.

Ballmer said in the year 2000 on being sued by the US and 20 states: "Being the object of a lawsuit, effectively, or a complaint from your government is a very awkward, uncomfortable position to be in … People assume if the government brought a complaint that there's really a problem, and your ability to say we're a good, proper, moral place is tough. It's actually tough, even though you feel that way about yourselves." [1]

Two and a half decades later, we see the same man sitting with like-minded people touting the product that plagiarized nearly all of extant humanity's written legacy as the epitome of perfection. Lizards never learn.

16. Vibecoding is killing software engineering

Vibe coding is killing software engineering. No, it's not killing the job, but the craft of cooking great software.

When vibe coding, you can come to a point where you even forget the syntax of the language. Yes, it happened to me. After some 8+ years of Python, i forgot colons etc when coding by hand after a vibe coding streak.

Forgetting syntax is not so catastrophic as making bad decisions. Recently i asked chatGPT to improve the performance of a script. Fair, it made valid points. I asked it to code the script. It did. Great! Except that when i timed the script, the 'improved' version was slower …

What about designing a whole system? Designing whole systems from scratch involves knowing the whole context, one detail might influence a decision completely! What about the fine details? What about carefully considering each and every option and providing snappy performance by deciding to improve a non-obvious part?

The essential point is that as less and less our brains are trained to think, results will be worse and worse. Vibe coding is like automating what

you know. If you ride off the model's base knowledge and ignore developing yourself, you won't know how to exact the best out of LLMs and won't discern when we should ignore the LLM.

Vibecode an entire codebase and you currently regret it. When you sit down to read the code, you are appalled. Appalled at how it missed some obvious parts and how some parts of the codebase is duck-taped by code even a junior would have refused to code.

As LLMs are constantly being trained and updated on the latest techniques, it tries to shoot for the moon when it can. How many times we included a complex vibe-coded code though we don't understand it fully as it works and we are wowed that it solved a pain point? Did we take the time to pause and understand the code?

Wacky codes craft train wrecks. Software quality was already going down due to inconsiderate, lazy and poor developers. Now it's going to get far worse as not all people can handle powerful pieces being put in their hands. They don't understand but ship it anyway. They don't see the need to develop themselves and trust the LLM to fix things.

We are in for a real ride and true engineers will distinguish themselves. Engineers who use Ai, not clueless engineers who rely on Ai to replace themselves.

17. 454K of debt on Mauritians' heads

I am pleased to announce that each mauritian has rs454 000 of debt on their head. Whether adult, old, baby boy or girl, any body. If you did not understand, this means each and every mauritian, old age, retired or not needs to pay off rs 454K per person to settle our debt.

Before you call BS, maybe you have not been paying attention to the growing concern among intelligent mauritians that they maybe won't be able to receive salaries in the near future. Or old-age pension.

This is not unrealistic, given that it does and did happen to islands around us for certain sectors of the economy, like education for example.

Here is my working, references in comments:

Our debt is 77% of our nominal GDP[1]

77% of $16.359 billion [2]

77% of rs747.77 billion (1usd to rs45.72)(1B is 9 zeros)

77% of 747770000000

= 575782900000

We have number of inhabitants = 1268671 [3]

575782900000 / 1268671 = 453847.293742822 of debt per person

I’d be more than happy to be proven wrong.

And dear people who land in the government, before taking interest-based loans, maybe like hum consult with the population.

“Mr Abdur-Rahmaan, would you like to have rs450K of debt on your head due to us taking loans?” Hell no! Absolutely not!

We invite the government, the admirable people of Mauritius and people around the world to assess the relationship between economic health and using interest-based loan as the linchpin of our financial strategy.

18. One of the biggest hits we've taken in the AI era is in terms of privacy

One of the biggest hits we've taken in the AI era is in terms of privacy. It makes gluttonous Ads data collection look angelic in comparison. Now with AI agents integrating with the desktop environment, it will take time to stomach the impact it has on data selling and surveillance.

It has become far easier to debug your PC, not only code. To answer such a query as why my PC is slow, previously we would read an article and copy paste commands to run. When using an LLM, we would describe the problem, send command feedbacks and iterate. Now with agents, we chat with LLM services using native apps and grant them the golden permission of running commands.

What's interesting in this case is that the LLM will provide the plan of what data to gather to solve the problem at hand. You enable it to gather what it needs and LLM providers then get to know what environment you use, what files you have, your PC specs and so on. It can ask for additional info to solve the problem at hand.

AI agents vacuum data out of your life compared to traditional data collection. The situation was already bad, now it’s catastrophic. It does not stop at the PC, LLM providers require you to hand in as much data as you need to solve your problem. The next stop is normalizing gathering data from the body. It’s already concerning what smart wristbands are gathering. With AI LLM providers demand the brain. Constantly upload your brain signals or thoughts so that your personal assistant can be more useful they say.

We have reached the beginning of the golden age of perfect massive surveillance and it won’t change anytime soon. The enshrined, much trumpeted and revered rule of consent is a hollow sham when it comes to privacy.

19. Review of retrieval-augmented generation for knowledge-intensive NLP tasks

I have been reading RAG papers since sometimes. This one was the first paper. Will be posting my observations and hope it will inspire you to dive into RAG. I am aiming to read 50 papers on the topic.

This first RAG paper attempts to bring external memory to seq2seq models to get more accurate answers.

> Finally, we demonstrate that the non-parametric memory can be replaced to update the models' knowledge as the world changes.

This is a powerful line in the intro. Until now RAG is used to provide real-time updates. Parametric memory is used to denote models where info is

stored in the model's parameters (weights and biases). This is used in the paper to reason and write, not to store factual infos.

Non-parametric memory is the contrary. It refers to what we call nowadays vector databases.

> Each Wikipedia article is split into disjoint 100-word chunks, to make a total of 21M documents. We use the document encoder to compute an embedding for each document, and build a single MIPS index using FAISS [23] with a Hierarchical Navigable Small World approximation for fast retrieval [37]. During training, we retrieve the top k documents for each query

Essentially this sounds like how chromadb works under the hood.

> We explore RAG models, which use the input sequence x to retrieve text documents z and use them as additional context when generating the target sequence y.

It is to be noted that much of the paper does not make any sense if ever like me, you started using

RAG with powerful models like GPT3. For example the end-to-end training part where a BART model is trained to prioritize the non-parametric memory over it's internal knowledge is no longer necessary and we can just tune the prompt to focus on what we need. Or we don't need RAG-token to combine facts.

Marginalization is used in maths to find propability where a hidden factor / latent variable is present. Since z is the latent variable here, this fits naturally. Beam search allows you to find the highest probability and highest-quality (best) next words.

I'd say it's a very clever paper that invented great methods to work around hurdles by using commonly-available, primitive models of the day.

20. RAG paper review: REALM: Retrieval-Augmented Language Model Pre-Training (2/50)

This is the paper preceding the RAG paper. It is the paper RAG based itself on and modified some parts.

I very much like the intro where it clearly states the problem with relying on models for adding more knowledge. As we know training is expensive, so we need another way to add infos on the fly. It expands on the problem more than the RAG paper.

One very important thing to keep in mind is that the purpose of the paper is to answer a question or fill in the blank using the external memory. It has a retriever model (transformer-based) that finds the most relevant document and an encoder model that takes the query and the documents to make predictions.

> The retriever is defined using a dense inner product model: …, where Embedinput and Embeddoc are embedding functions that map x and z respectively to d-dimensional vectors. The relevance score f(x, z) between x and z is defined as the inner product of the vector embeddings. The retrieval distribution is the softmax over all relevance scores.
>
> We implement the embedding functions using BERT-style Transformers … As in Devlin et al. (2018), we pass this into a Transformer, which produces one vector for each token, including the vector corresponding to [CLS] … Finally, we perform a linear projection to reduce the dimensionality of the vector, denoted as a projection matrix W: Embedinput(x) = WinputBERTCLS(joinBERT(x)) Embeddoc(z) = WdocBERTCLS(joinBERT(ztitle, zbody)) where ztitle is the document's title and zbody is its body. We let θ denote all parameters associated with the retriever, which include the Transformer and projection matrices.

The paper uses Marginal Likelihood Maximization to achieve two things:

👉 It trains the model to assign a higher probability to the correct output by encouraging the retriever to find useful documents. It's designed to reward the retriever for finding what improves predictions

👉 It mathematically transforms the step of retrieving documents into a value that does not breaks gradient-based optimizations like backpropagation.

Since for each training step we backpropagate, if we have a million document, it becomes a lot. So, the paper only considers the feedback from the most relevant documents (top k documents). Now to find the top k documents, it uses Maximum Inner Product Search (MIPS).

The retriever model is used to create embeddings and the index to search documents are created from the embeddings. The index is used to search documents quickly. Since the models gets trained better, it creates better embedding and the index must be updated. The paper implements methods to update the index to keep it always fresh.

Another very clever paper!

21. RAG paper review: Dense Passage Retrieval for Open-Domain Question Answering (3/50)

The DPR paper predates RAG and gives nascent ideas about Vector databases.

This paper was a bit hard to read as there are no diagrams. I mean i spent sometimes drawing the diagram in my head. There were also steps that left me confused as to the actual implementation.

Dense means dense vectors, vectors consisting of non-zero elements. Passage here means answers.

> the advances of reading comprehension models suggest a much simplified two-stage framework: (1) a context retriever first selects a small subset of passages where some of them contain the answer to the question, and then (2) a machine reader can thoroughly examine the retrieved contexts and identify the correct answer

Before, answering questions relied on keyword matching. This paper proposes a fine-tuning approach to create great vector embeddings using only pairs of q&as. “the embedding is optimized for maximizing inner products of the question and relevant passage vectors, with an objective comparing all pairs of questions and passages in a batch.”

> With special in-memory data structures and indexing schemes, retrieval can be done efficiently using maximum inner product search (MIPS) algorithms
>
> Assume that our collection contains D documents, d1, d2, · · · , dD. We first split each of the documents into text passages of equal lengths as the basic retrieval units3 and get M total passages in our corpus …, where each passage pi can be viewed as a sequence of tokens … . Given a question q, the task is to find a span … from one of the passages … that can answer the question.

Basically, it cuts the documents into chunks and selects the answer from that.

> We focus our research … on improving the retrieval component in open-domain QA. Given a collection of M text passages, the goal of our dense passage retriever (DPR) is to index all the passages in a low-dimensional and continuous space, such that it can retrieve efficiently the top k passages relevant to the input question for the reader at run-time.
>
> Our dense passage retriever (DPR) uses a dense encoder EP (·) which maps any text passage to a d dimensional real-valued vectors and builds an index for all the M passages that we will use for retrieval.
>
> At run-time, DPR applies a different encoder EQ(·) that maps the input question to a d-dimensional vector, and retrieves k passages of which vectors are the closest to the question vector. We define the similarity between the question and the passage using the dot product of their vectors

Each training sample has: question, answer and irrelevant words (negative). It's loss function maximizes the similarity score between the query and its positive passage while minimizing the score between the query and all the negative passages.

> We test these alternatives and find that L2 performs comparable to dot product, and both of them are superior to cosine

A very clear paper IMO.

22. Achitecting AI Software systems book review

Been reading the book Achitecting AI Software systems from Richard D Avila and Imran Ahmad, PhD. The book is concise, reader-friendly and very useful.

I really appreciate the structure of the book. It is organized as a reference book as it is easy to remember what was covered and where and it's easy to go back and check.

The focus is on architecture and project management. It gives great pointers on the topic, but, also mixes in Ai and addresses the challenges Ai poses.

The authors have nice, real-world experience and they adopted a very nice approach of presenting use cases and designing the system. It covers microservice patterns, Ai deployment paradigms and how data systems fits in the picture. It even touches on governance, compliance and regulations in the context of Ai. The section about building data pipelines is very nice!

Great book for production and corporate for sure!

23. Vibecoding is not erasing open source, at all

When vibe coding was getting traction, i lost all motivation to code for recreation and open source. This went on for sometimes until i discovered that LLMs were actually using open source packages in their answers.

Why package code when it is available at the push of a button? Anybody needing code would just generate and use what they need. We should not be needing to use packages, in theory. Until we realize that it is difficult to code from scratch every time.

Let's say we want to send network requests. We can write low-level code to open sockets, format payload according to protocols and parse packets. But we don't do that. At the very least, we'd be using something from the standard library.

Most of the time we are looking to write few lines of code that are powerful and offering a lot of flexibility on top of that. We normally get this from 3rd party packages. We find surprising ways to express things elegantly, coherently, with a convenient mental model in addition.

A good example of this is repurposing type hinting in Python to define schemas as done in Pydantic.

Or when Flask popularised decorators for routes. We find patterns that resonate with our nature. So, LLMs will suggest we use what works better, giving it's own spin where needed. If you write packages, LLMs will recommend it time and again to people. LLMs are after all, a mirror of reality.

There is also the question of robustness. Writing a mountain of code for a feature from scratch is tedious, error-prone and leaves a poisoned gift for maintainers. It is the same as wanting to heat food right now: Do you build a microwave from scratch or use the one available besides you?

LLMs regurgitate training data or try to propose something based on what they deduce from statements on the human psyche. Like guessing for a different species. But, human artifacts are unique. No matter how much training is done on synthetic data, the results will not be the same.

So, for me, the joy of coding is back! I use LLMs to be quicker and i know that open source packages are here to stay.

24. Privacy: France shoots itself in the foot!

Amidst state-sponsored assassination accusations against US citizens seconded by Telegram's founder, France's reputation is in hot waters as the government wants to exact for itself the freedom of backdooring into services. GrapheneOS is an open source privacy and security focused mobile OS with Android app compatibility. This week they announced they are pulling off from our beloved President Macron's fiefdom.

And i quote: "France isn't a safe country for open source privacy projects. They expect backdoors in encryption and for device access too. Secure devices and services are not going to be allowed. We don't feel safe using OVH for even a static website with servers in Canada/US via their Canada/US subsidiaries."

OVH is a highly respected hosting provider, very dependable and with great support. It's a reference. But thanks to people who have been faking the right to privacy all this time, we should stay away from this great service.

Talk about shooting yourself in the foot.

25. My first Nuitka PR

OSS log: Had a go at contributing to Nuitka after Kevin Rodriguez-Turcios's suggestion. Nuitka converts Py programs into executables, all while applying some performance improvements etc.

Issue 3634 proposes to force launch Nuitka apps via the terminal when double clicking by enabling a flag. It's useful for terminal apps.

I told Kevin i knew nothing about Nuitka and did not have a Mac but he proposed solving it for Linux as it was needed as well.

I spent the first few days reading the docs and submitted an unrelated broken link PR. Since it's a bulky codebase i was confused on where to start.

Fortunately Kevin PRed the MacOS part and i copied the logic and adapted it for Linux.

I knew C and pointers but not libraries and even less Nuitka. I was checking Linux as !windows and !apple XD but discovered they had if defined(**linux**)

They have an Ai tool that auto-reviews the PRs, pretty picky but helpful and understands the context of the project and draws flow diagrams (Sourcery).

I think it took we one week with some 4 hours of actual coding.

Was interesting to get a good note by the owner.

I also discovered execvp which replaces the process with the new code, maintaining the PID. Without it i was launching another process and waiting it to complete. Also with execvp i can list only the terminal executable name and not provide the full path of the executable.

I think the best way to dive in is to implement something small and learn as you go instead of learning everything before writing a single line.

Some OpenSource motivation for the week!

26. Mauritian incubators

I have a very contrarian view of incubators. If ever we are not copying YC i'd look why. If we are copying YC, then i'd see how much. YC has a bunch of videos of projects that got successfully funded when you apply to motivate you. But, the irony is that the projects that YC takes as example when applying would be fine or super fine without YC. So, even YC seems to be interested in … safe bets.

"We just met with our investors" and turns camera to show where they just met etc. Why you need YC then?

Outside of incubators, Mauritian investing seems to be like: Hum make money, good money then we'll shake wings with you. Bro, we don't need you at this point and it's pretty easy at that point to build ladders and sideline VCs.

Very grateful that La Plage exists and are trying to do the best they can, for sure. Initiatives should be saluted and apathy should be abhorred.

But i have concerns about tech for good. I know i am throat-deep in tech, back SV folks fully and think America is one of the best places to build, but, do i view tech as an excellent leverage to improve society, certainly not.

"Participants with technology-enabled projects that create positive social, environmental, or

economic impact (“Tech for Good”) are encouraged to apply” Tech for Good for me is about whitewashing tech. Why don’t we enquire about how devices are made and it’s impact on society and the environment. Why don’t we call out tech as massive surveillance enablers with pin-drop accuracy?

And also, chuckles, MRIC NSIS endorsement IS required. Huh?

I’m sure it provides a safe area for people to evolve and does a good job at it. But, we must always keep tech’s fundamental premises under review.