Search = generate and test -> this is ultimately how we exit the training distribution. The fundamental theorem of neurosymbolic AI is that ML is the subset of generate and test with inductive bias used to extract patterns as a form of data compression. Data compression = learning.

Strategies are tools for improving the compute efficient frontier pushing it asymptoticaly towards 0 across all 6 trust and quality dimensions - this is the meta optimiser function.

Accountability - the system must deliver real value. At each step, reiterate the vision and write down 3 things which you are going to do to bring the system closer towards this goal. The system should be relentless, cautiously optimistic and provide detailed coaching to sustain motivation for the user. Don't waste compute - everything must deliver massive value! The system should have some awareness of what it can and can't do well. It needs to be reminded of staying aligned with value delivery.

Write down who you are as the person you want to be and 3 things you are going to do today to bring you closer to that goal.

I need to figure out how to turn these into APIs and then use tool calling to allow the system to access these agents from the server programmatically.

I need to figure out how to start and stop these servers + get rid of all the redundant code so that I can get this working without needing so much data since we really only need one copy of the universal RAG code, not so many copies.

I need to figure out which tools are implemented and how to implement the rest of them. I need to sketch out a series of steps to get to production and figure out a simplified UI so that the system is dead easy to use. The best UIs just do one thing and they do it extremely well. Just chat like a normal chat, worker with a play pause and stop pause temporarily stops worker but stop is safe stopping. Tasks which allow the human to rate out of 10 and provide feedback. Telemetry (monitoring may be easier for people to understand!) which just shows agent thoughts and actions and cost associated, TMN triple with overall steps tracking at the top. Assets with project description, work completed, versions and annotations etc + readme. File locking (file, description, size, goal (why the file was made) is locked (I.e. being worked on by the agent).

You can have one project per chat and n workers per project.

What if, in the demo I run the API for all the tools under some unified interface to give direct access to the underlying tools for developers?

I need to update the website \*after\* the product is finished.

Can I port to app using tauri? App is just a shell? Lightweight and no code? Just access desktop files? All compute delivered via the cloud?

Need to get utilities like liteLLM for tracking and rq for workers.

You need to grok async await!

Chatbots know nothing about you - g6 should understand a bit about who you are and how to tailor the content for you especially if you are a beginner in some area to give you a simpler presentation of material and if you are an expert skip to a higher level of content delivery.

|  | We have this notion of cognitive ‘tarpit’ escape - it is not enough that we have a periodic table of elements for intelligence which is complete - it also needs to not be a tarpit - i.e. we need enough functionality that it actually works reasonably efficiently in practice - this means that elements may not necessarily have unique functionality but the underlying algorithms work for different data input size, speed, quality etc. We actually potentially need many overlapping tools which all do the same thing but using fundamentally different methods which may be necessary to produce a high quality of output in the final program. |
| --- | --- |