Considering vector database?

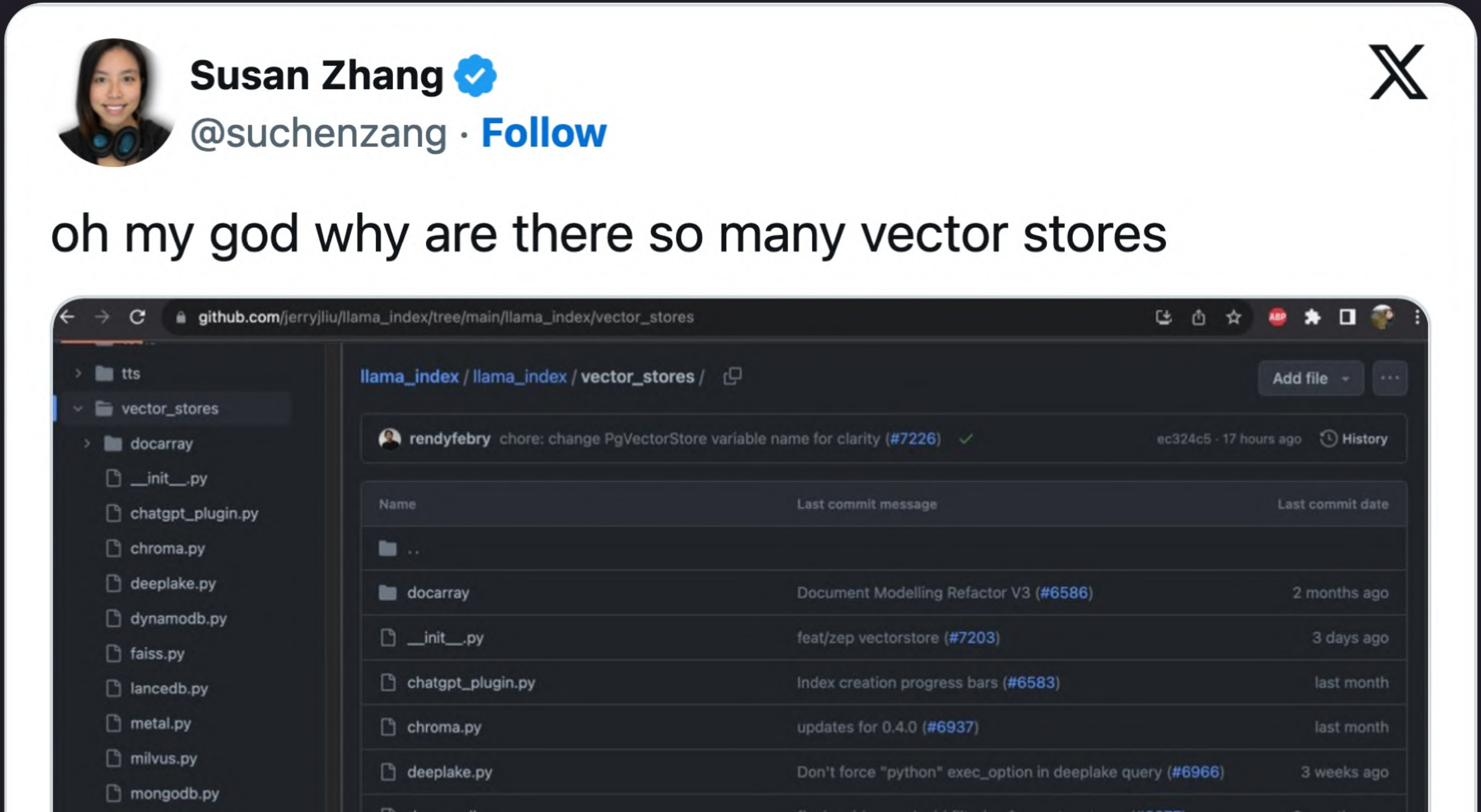
You don't need it. 🙅

Guide for **every CTO** & **engineers**

vec3.ai

by Dariusz Semba

Currently, there are over 20 vector databases on the market!


Susan Zhang
@suchenzang · Follow

oh my god why are there so many vector stores

github.com/jerryjliu/llama_index/tree/main/llama_index/vector_stores

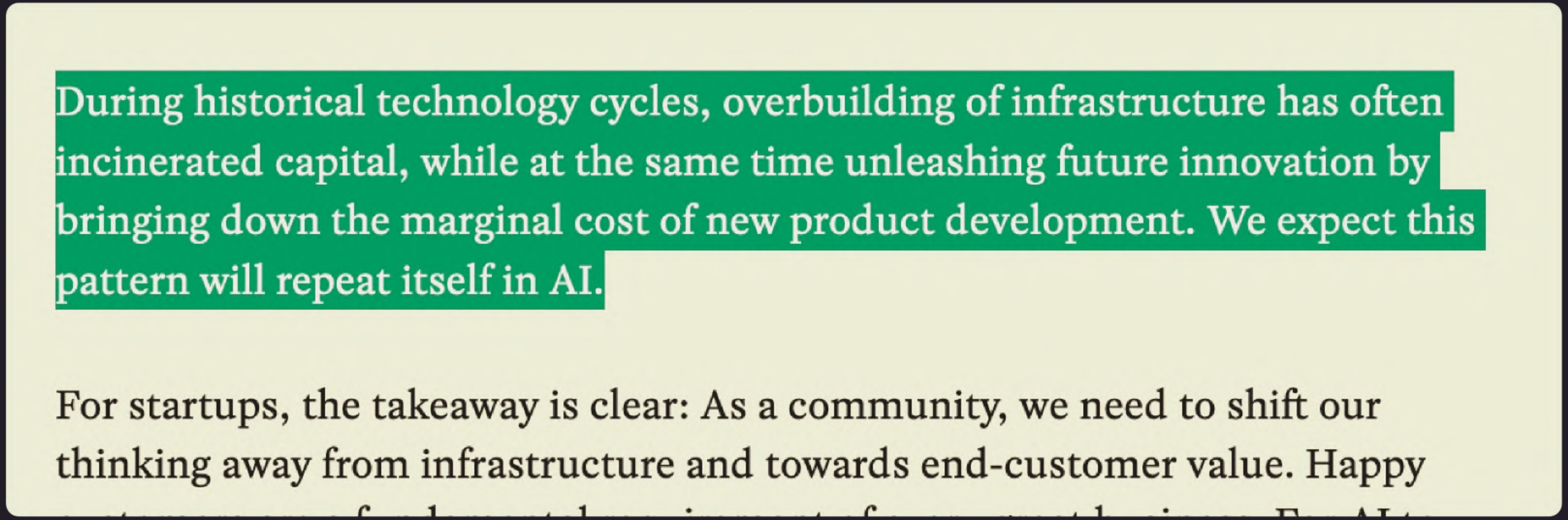
- tts
- vector_stores
  - docarray
  - __init__.py
  - chatgpt_plugin.py
  - chroma.py
  - deeplake.py
  - dynamodb.py
  - faiss.py
  - lancedb.py
  - metal.py
  - milvus.py
  - mongodb.py

llama_index / llama_index / vector_stores /

rendyfebry chore: change PgVectorStore variable name for clarity (#7226) ec324c5 · 17 hours ago History

Name	Last commit message	Last commit date
..		
docarray	Document Modelling Refactor V3 (#6586)	2 months ago
__init__.py	feat/zep vectorstore (#7203)	3 days ago
chatgpt_plugin.py	Index creation progress bars (#6583)	last month
chroma.py	updates for 0.4.0 (#6937)	last month
deeplake.py	Don't force "python" exec_option in deeplake query (#6966)	3 weeks ago

Even VCs like Sequoia admit:
infrastructure is getting overbuilt.



During historical technology cycles, overbuilding of infrastructure has often incinerated capital, while at the same time unleashing future innovation by bringing down the marginal cost of new product development. We expect this pattern will repeat itself in AI.

For startups, the takeaway is clear: As a community, we need to shift our thinking away from infrastructure and towards end-customer value. Happy

From AI's $200B Question

by David Cahn, Sequoia's blog

They see this with GPUs - vector dbs are much easier to build in comparison

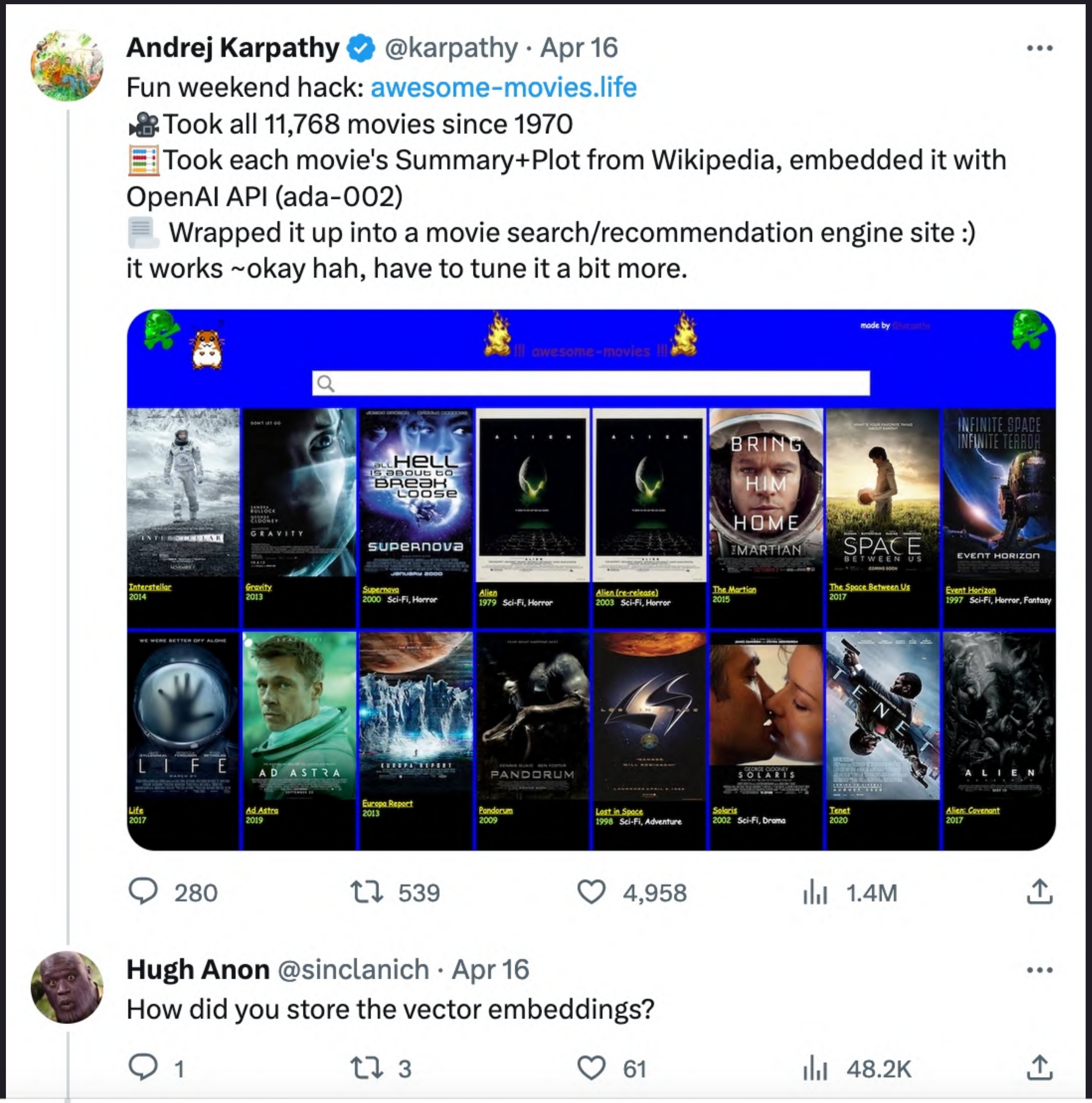
VCs see vector databases as an investment in [*"picks and shovels"*] for AI, with a proven business model (database).

💸

With so much funding (thus sponsored content) and with all the hype around AI - it's easy to assume you need a vector db in your AI project.

Why you don’t need a vector db:

1. Traditional keyword search is good enough or will even better suit your needs
2. You don't have enough data to use it anyway (*see next slides*)
3. Information retrieval is not your core focus, and it’s better to integrate with out-of-the-box solution



Andrej Karpathy @karpathy · Apr 16

Fun weekend hack: awesome-movies.life

🎥Took all 11,768 movies since 1970

🧮Took each movie's Summary+Plot from Wikipedia, embedded it with OpenAI API (ada-002)

📃 Wrapped it up into a movie search/recommendation engine site :)

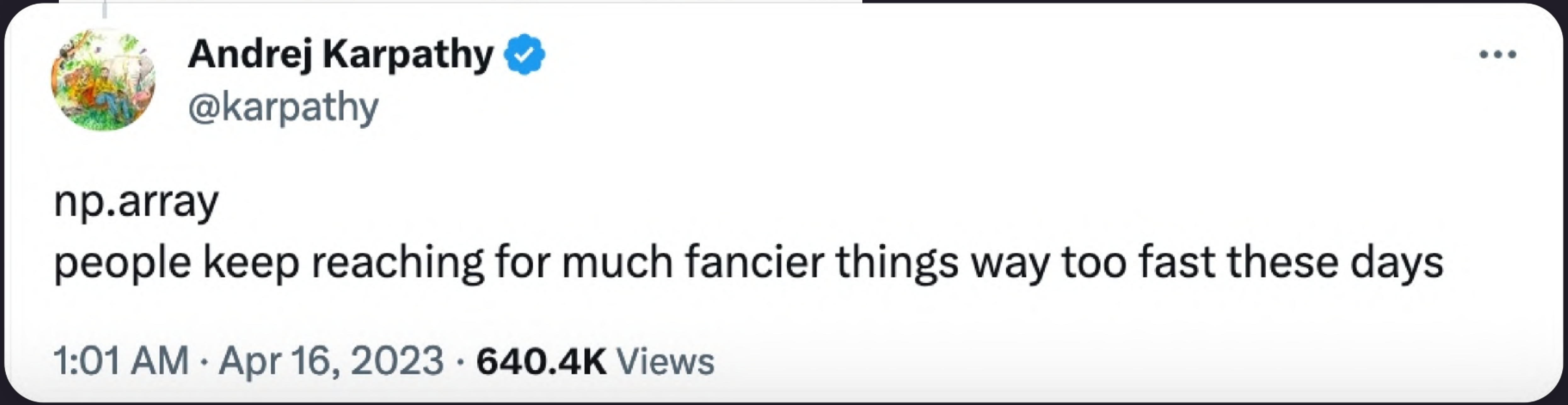
it works ~okay hah, have to tune it a bit more.

280 539 4,958 1.4M

Hugh Anon @sinclanich · Apr 16

How did you store the vector embeddings?

1 3 61 48.2K



Andrej Karpathy @karpathy

np.array

people keep reaching for much fancier things way too fast these days

1:01 AM · Apr 16, 2023 · 640.4K Views

Way too fast...

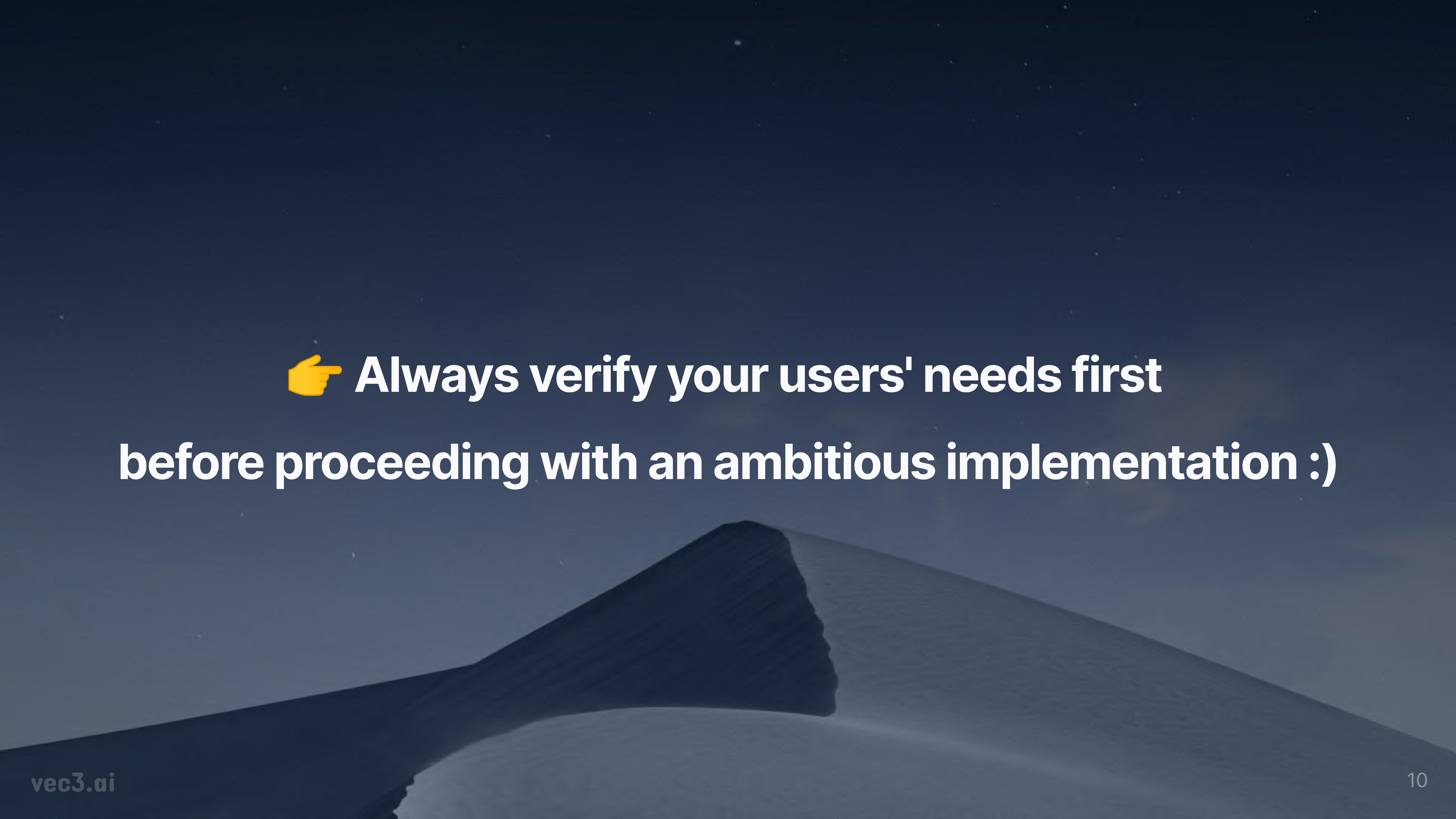
Alternatives sufficient for most data needs:

1. LLM alone can fit all your data in - no need for vector search
2. Exhaustive vector search (brute-force)
   - if you filter results first, there's fewer data to compare
   - can boost keyword search if done as a reranking step = hybrid search 🔥
3. Library for ANN (approximate nearest neighbors) search, e.g. FAISS
4. Your current database supports vector search efficiently enough

Vector database comes with a cost.

Hidden costs of vector databases:

1. Yet another database to maintain
2. Need to sync data with other dbs
3. Large memory overhead (or simply cost)
4. Need to train a custom embedding model for your data (and then again when the data changes)
5. Need to recompute embeddings when model changes - additional cost



👉 Always verify your users' needs first before proceeding with an ambitious implementation :)

Vector search is an optimization.

As engineers often say “*premature optimization is* ***the root of all evil***”.

What vector search tries to optimize

🔍 keyword search

✅ simpler, cheaper, well-known, interpretable & customizable

❌ doesn't capture semantics

🧠 LLMs

✅ capture semantics and conduct complex reasoning

❌ expensive, high latency

Vector search and keyword search are different capabilities

- Keyword search matches exact terms.
- Vector search captures semantic similarity.

Hence, vector search **doesn't exactly replace** keyword search.

Why keyword search rocks! 🎸

1. Often performs better than vector search
2. Generalizes well to unseen, out-of-domain data
3. Search mechanics:
    a. narrows down search results when query gets more specific
    b. efficient autocomplete capability
    c. highlights query matches
4. Easily interpretable, cheap, well-known

Vector search needs an embedding model

Vector search **relies on a neural net** that encodes the data into vectors:

- a generic model can perform worse on data from a narrow domain
- the model might drift over time, losing accuracy
- embedding model has its own "knowledge cutoff date"

“ **Vector search usually works better on demo (open domain) than real enterprise applications (closed domain)** ”

- Colin Harman, Head of Technology @ Nesh

Embeddings are inherently limited

1. Meaning squashed into a **limited-size vector**
2. Query and document interaction limited by a relatively low-dimensional vector **“dot product”** operation
3. Embedding models are **much smaller** (=less powerful) than LLMs
4. **Single time step** calculation - LLMs can do more complex reasoning when generating tokens

Most advanced solutions usually combine different techniques

keyword search + LLMs + vector search

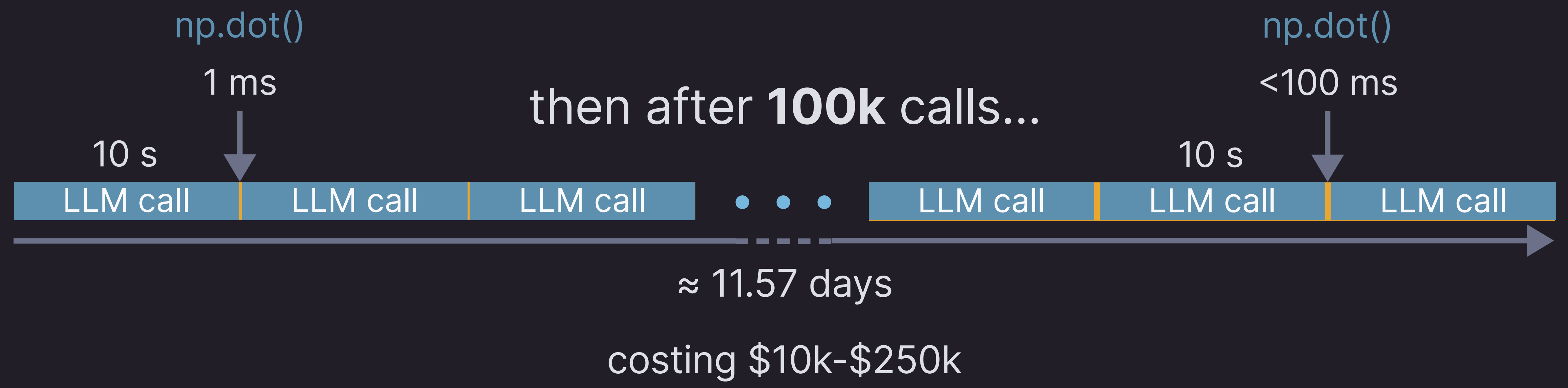
= **HYBRID SEARCH**

1. Make sure to start simple, with the right components.
2. Focus on optimizing whatever provides the best boost to overall accuracy.

Vector dbs x AutoGPT: an overkill solution

Let's say LLM call takes 10 seconds,
1 embedding is generated every 10s.

You reach 100k embeddings after 11 days.
Even then, brute-force vector search (np.dot) takes milliseconds



Embeddings	np.dot time	AutoGPT time	AutoGPT cost
1	<1ms	10s	$0.27
10k	<10ms	27h	$2.7k
100k	66ms	11d	$27k
300k	0.2s	34d	$81k
500k	5s	56d	$135k

No need for approximate nearest neighbors, let alone **vector databases**!

Optimizing LLM calls and accuracy of the system is far more important.

vec3.ai

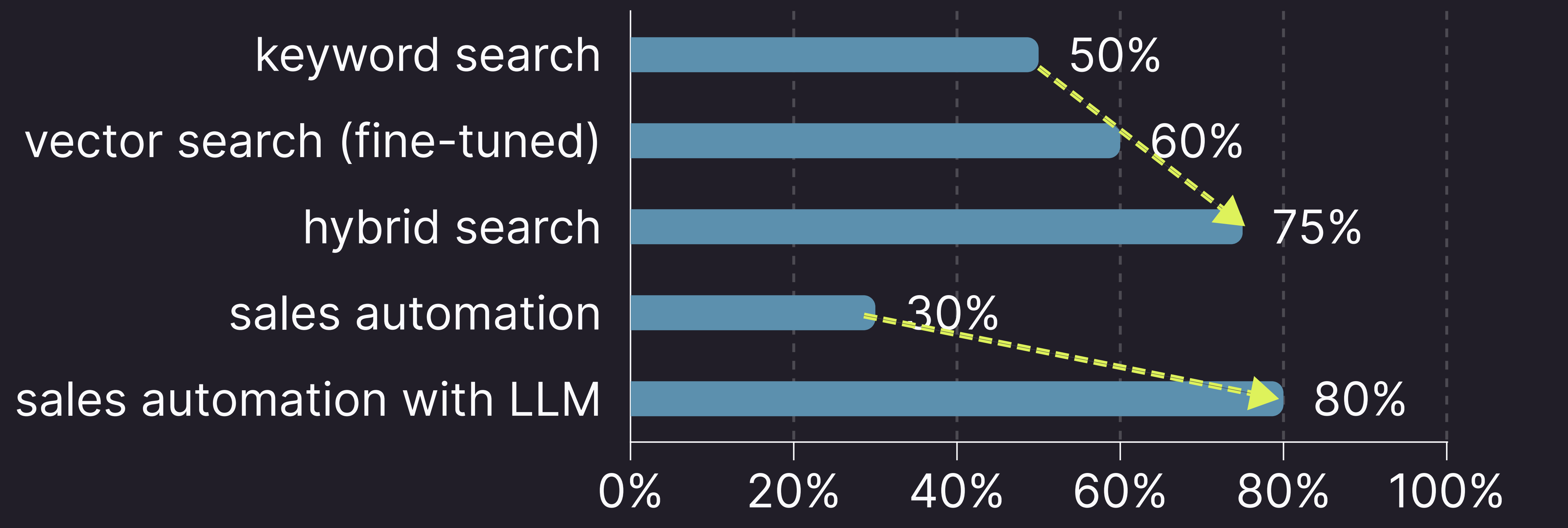
AI strategy

Focus on what brings most value to your users:

ChatGPT and its **RLHF** technique were a large breakthrough.
Vector search didn't have the same single "wow" moment.

Startups usually build value where no one else already had (blue ocean strategy).
Most novel value can be added through adopting **generative LLMs**.

AI strategy - seek 10x improvement



% of automation, made-up data serving as an example

In the example, improving company search with vector search yields 25 pp gain.

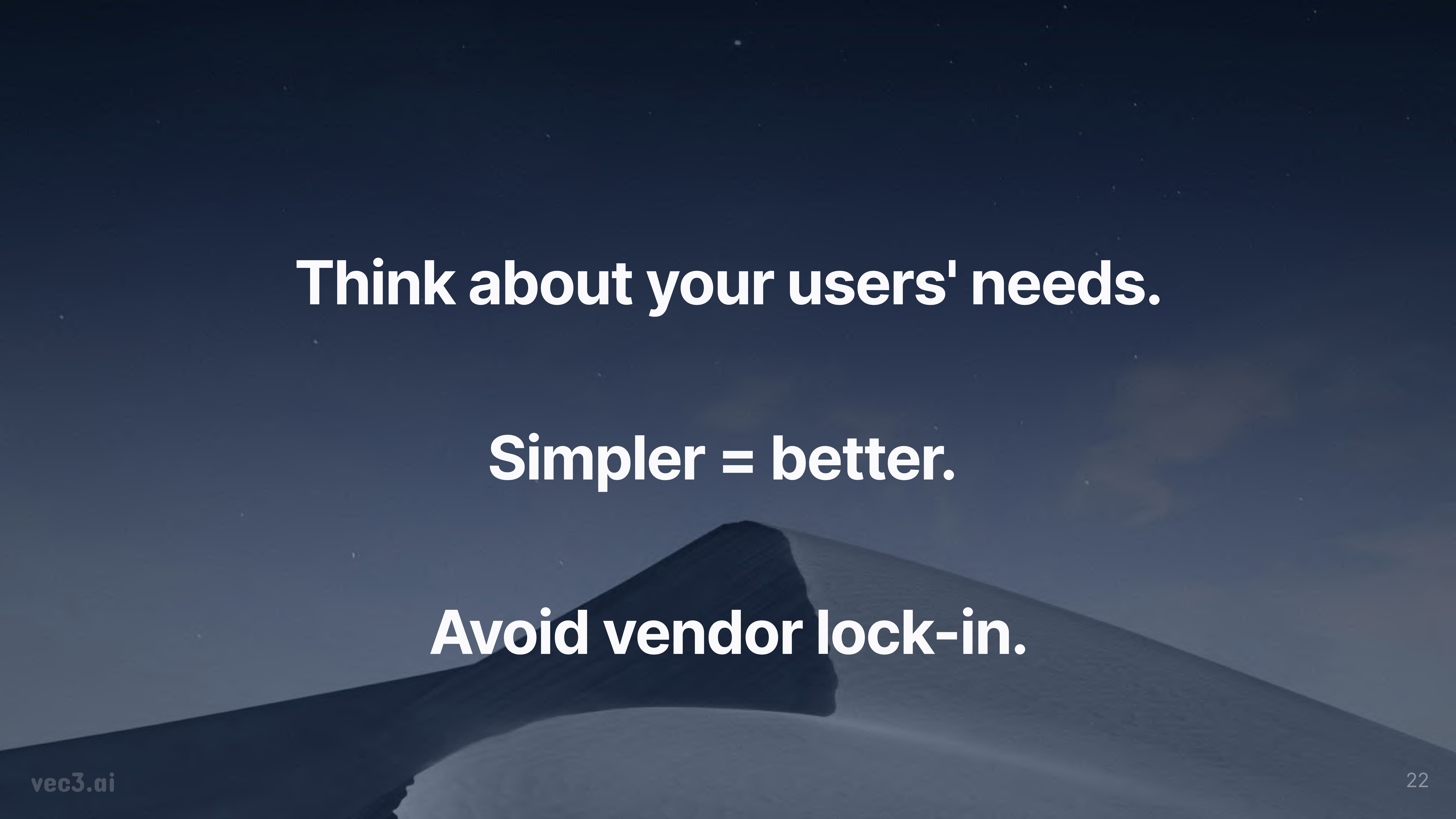
Building sales automation with LLMs would be much more disruptive compared to previous methods.

Some use cases might benefit much more from the current LLM revolution.

Future research 🔬

There's only so much meaning you can squeeze into a **vector**.
On the other hand, **generative LLMs** will keep getting better.

ChatGPT can already continuously query keyword search,
until it finds the right answer.
In the future of AI agents, that might actually be **the preferred way** of implementing search.



Think about your users' needs.

Simpler = better.

Avoid vendor lock-in.

If you found this page helpful, go ahead and share it with friends.

Let's keep AI efforts sane together :)

@Dariusz Semba

SOFTWISE.AI
The AI strike team

Sources / further reading

1. Vector Search with OpenAI Embeddings: Lucene Is All You Need paper
2. SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking paper
3. On Hybrid Search by Qdrant
4. Beware Tunnel Vision in AI Retrieval by Colin Harman
5. Emerging Architectures for LLM Applications by a16z
6. AI's $200B Question by Sequoia
7. Auto-GPT Unmasked: The Hype and Hard Truths of Its Production Pitfalls by Jina.AI
8. Why AutoGPT engineers ditched vector databases by Dariusz Semba
9. Introducing Natural Language Search for Podcast Episodes by Spotify
10. Why You Shouldn't Invest In Vector Databases? by Yingjun Wu