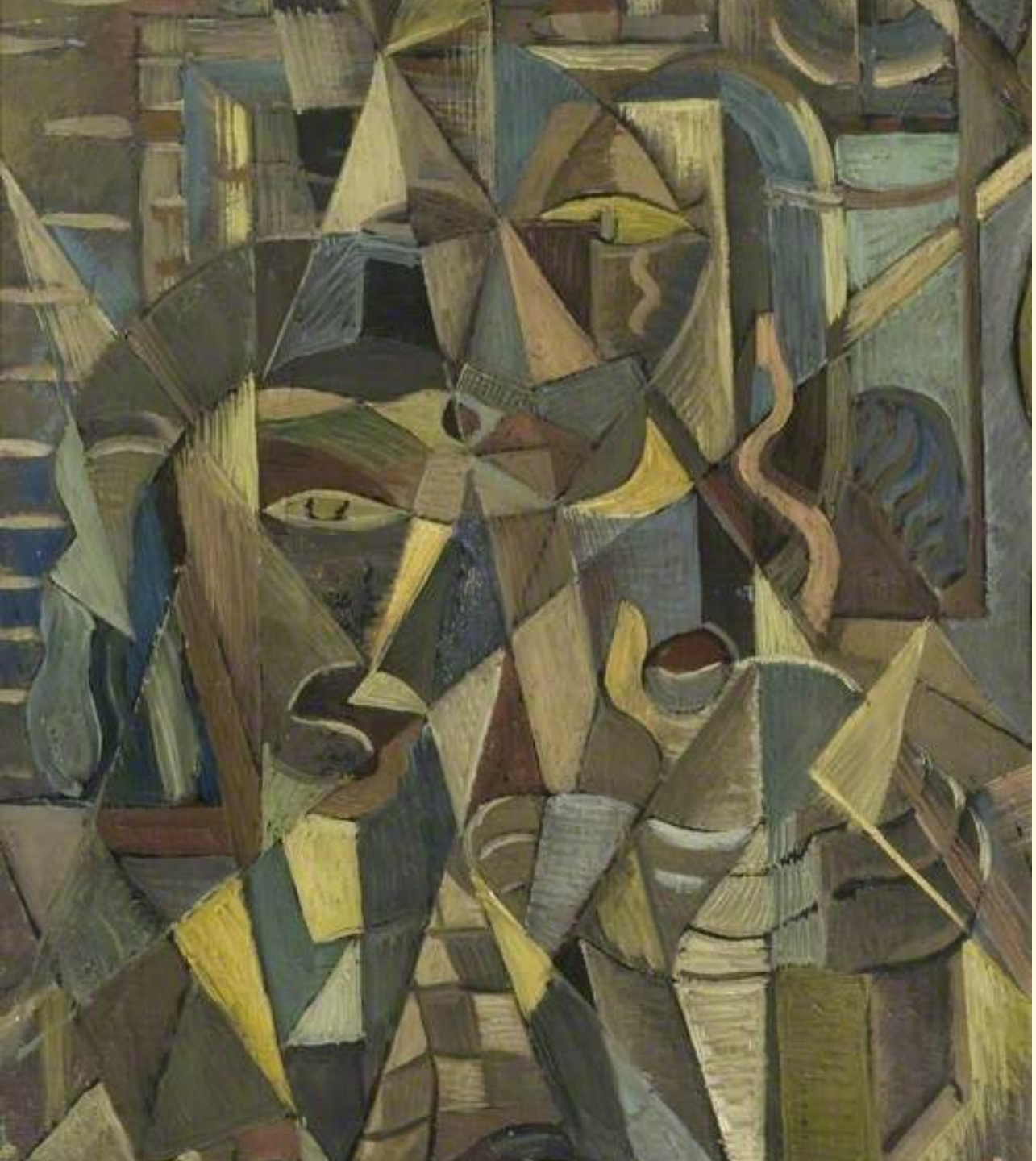


Scaling Retrieval using Rust

1.2B Chunks at <100ms

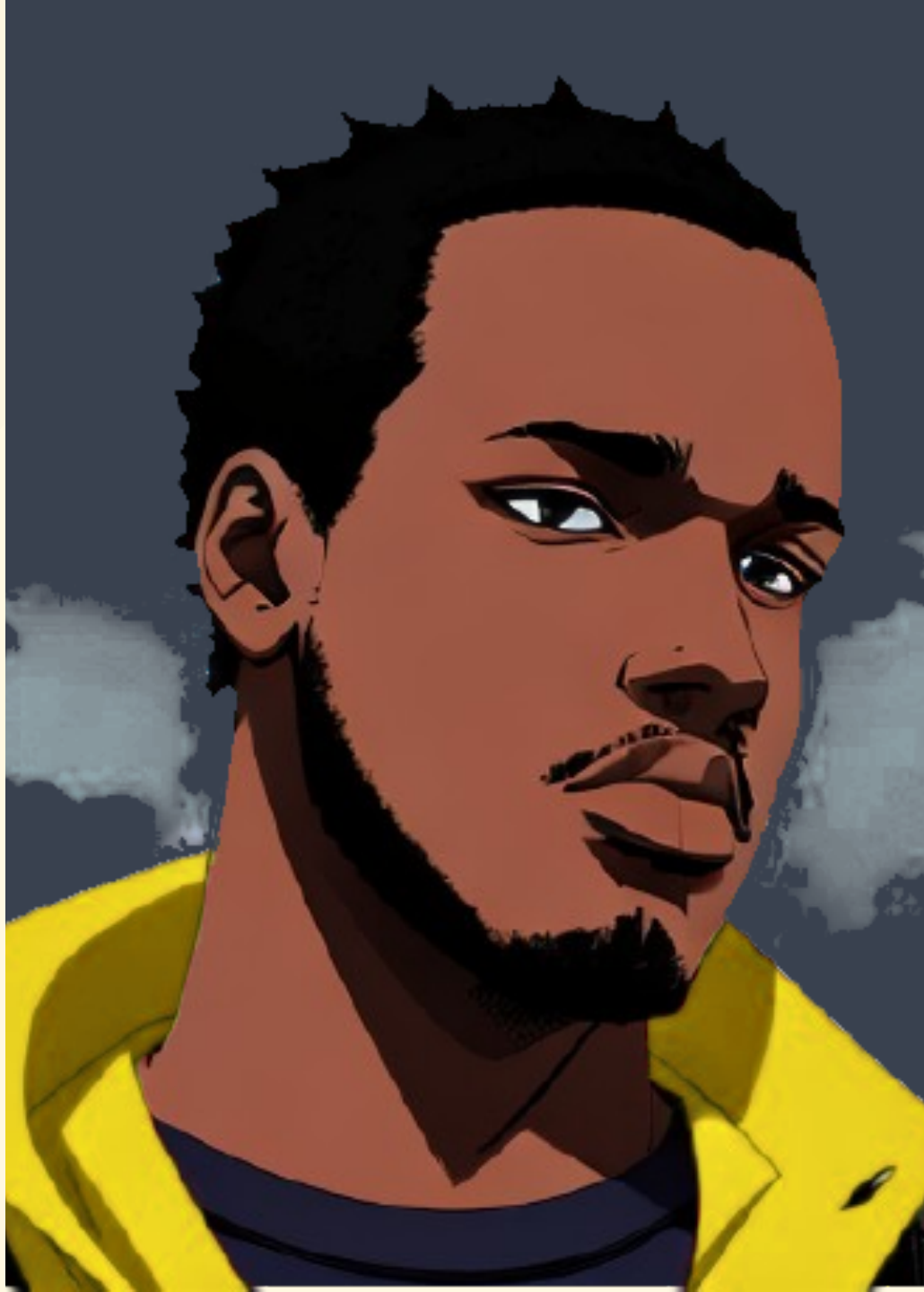
https://github.com/cdxker/scaling-search

Denzell Ford

Founder/CTO

trieve

Twitter: @cdxker
LinkedIn: - denzell-ford
GitHub: github.com/cdxker



Agenda

- Introduction
- Ingestion
  - Finding Bottlenecks
- Explainable Search Results
  - Fast Typo Correction
  - Highlights
- Sharding your search index

@cdxker - denzell.f@trieve.ai

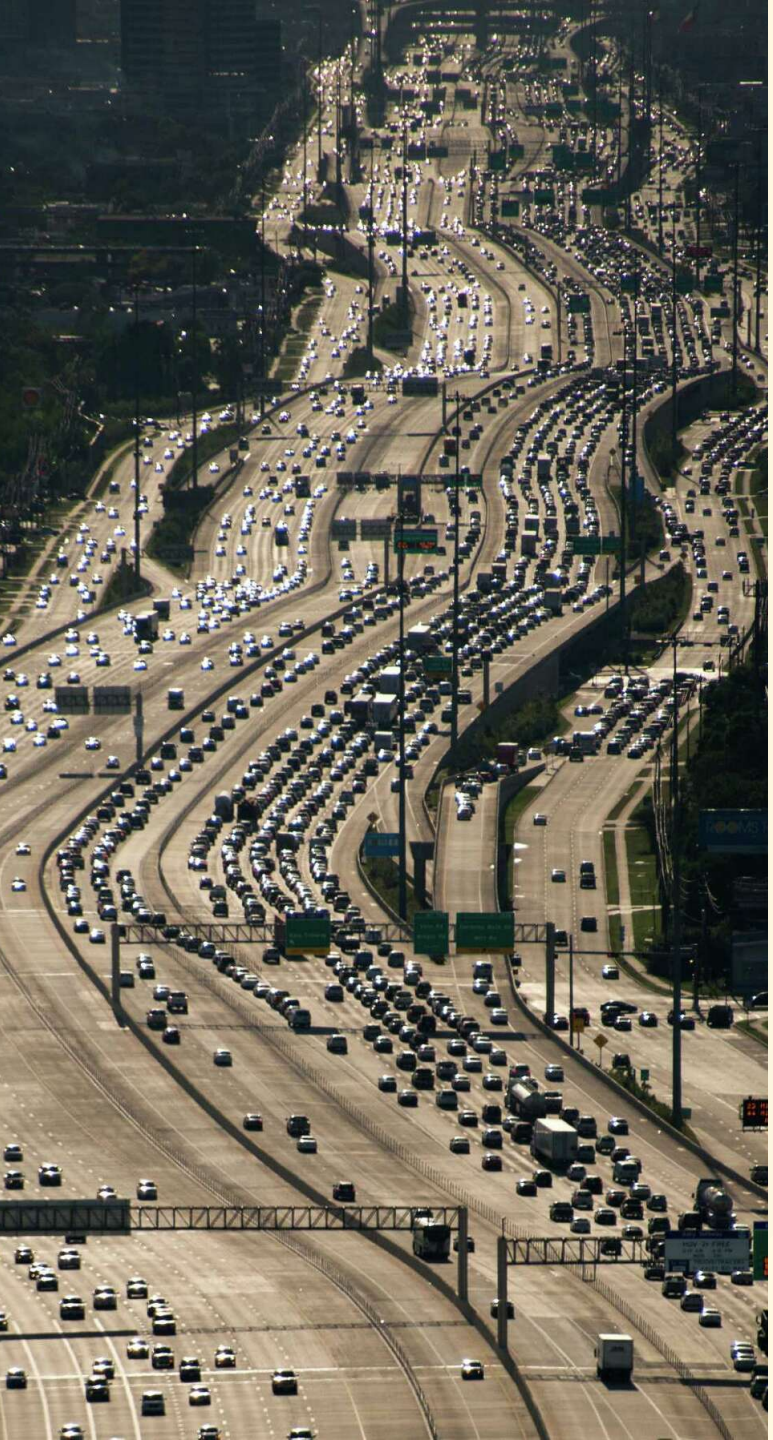
Introduction

def: AI Search is the use of vector embeddings to search for N similar items to a given query.

AI Search introduces a new set of challenges:

1. Efficient Ingestion.
2. Low latency retrieval.
3. Relevant and explainable search results.
4. Scalability.

@cdxker - denzell.f@trieve.ai

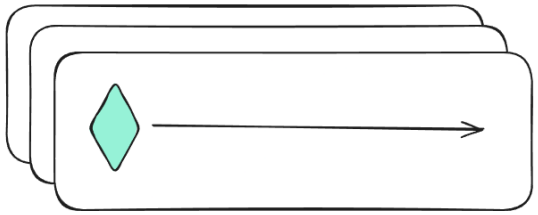


Why ingestion is hard

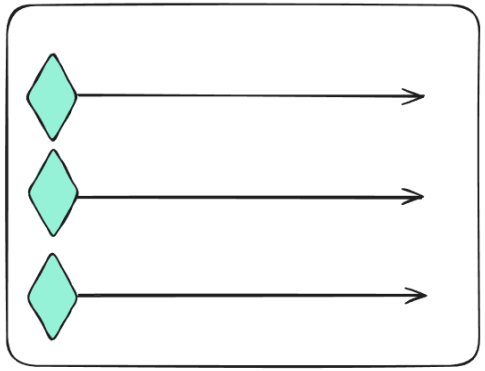
- Vector DB's are **not** the Primary DB. (no explicit relations)
- Calculating embeddings are expensive.
- Finding Bottle necks is hard.
- Iteration cycle is slower.

github.com/trieve/ingestion-worker

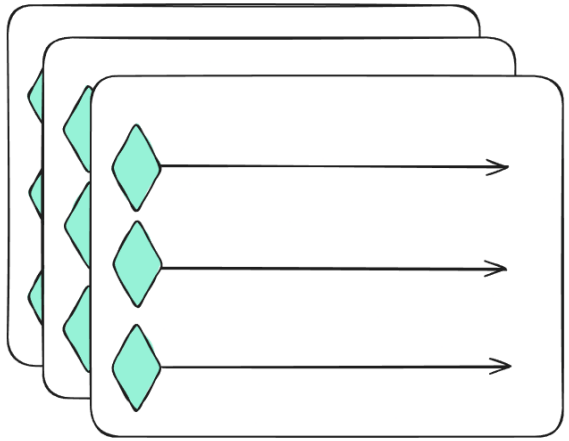
Horizontal



Vertical



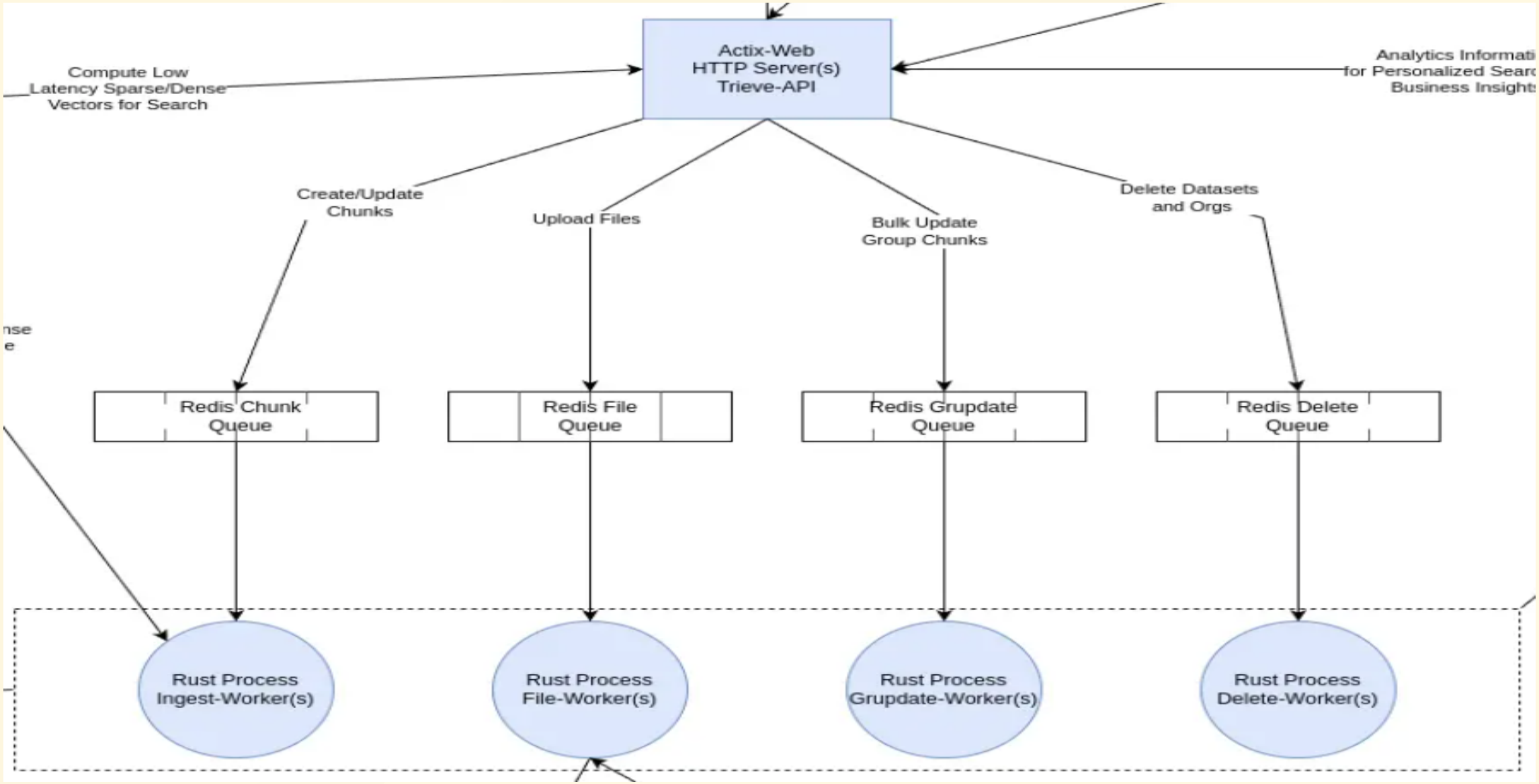
Hybrid



Ingestion Frameworks

- Single Thread, scale processes. (*horizontal scaling*)
- Multi Thread, scale threads. (*vertical scaling*)
- Multi Thread, scale threads **and** processes. (*hybrid scaling*)

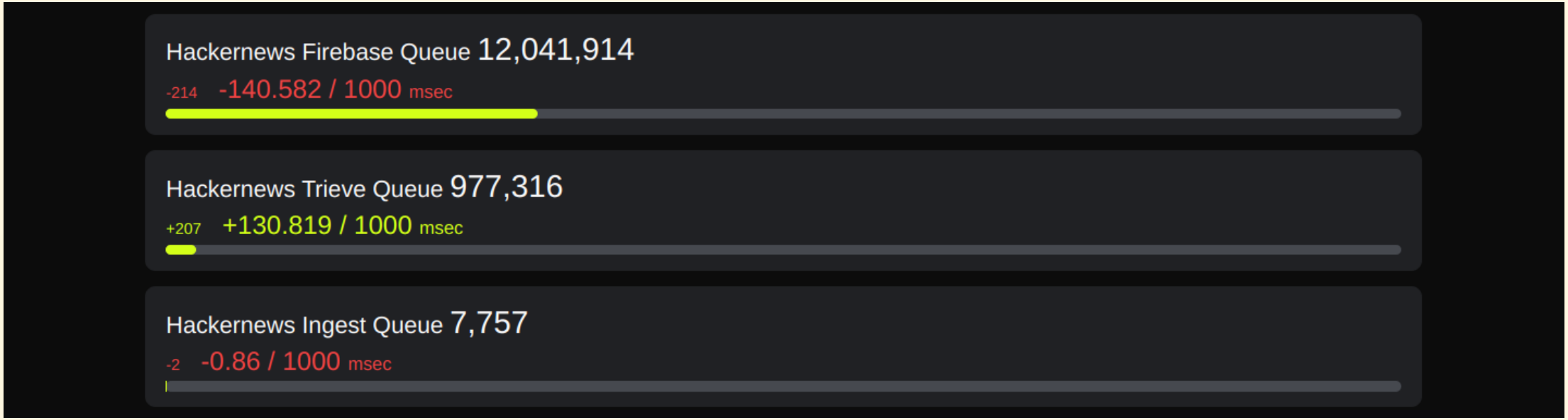
@cdxker - denzell.f@trieve.ai



@cdxker - denzell.f@trieve.ai

Finding Ingestion Bottlenecks

- Need to profile queue size changes



- Need to profile individual spans

@cdxker - denzell.f@trieve.ai

Bottlenecks

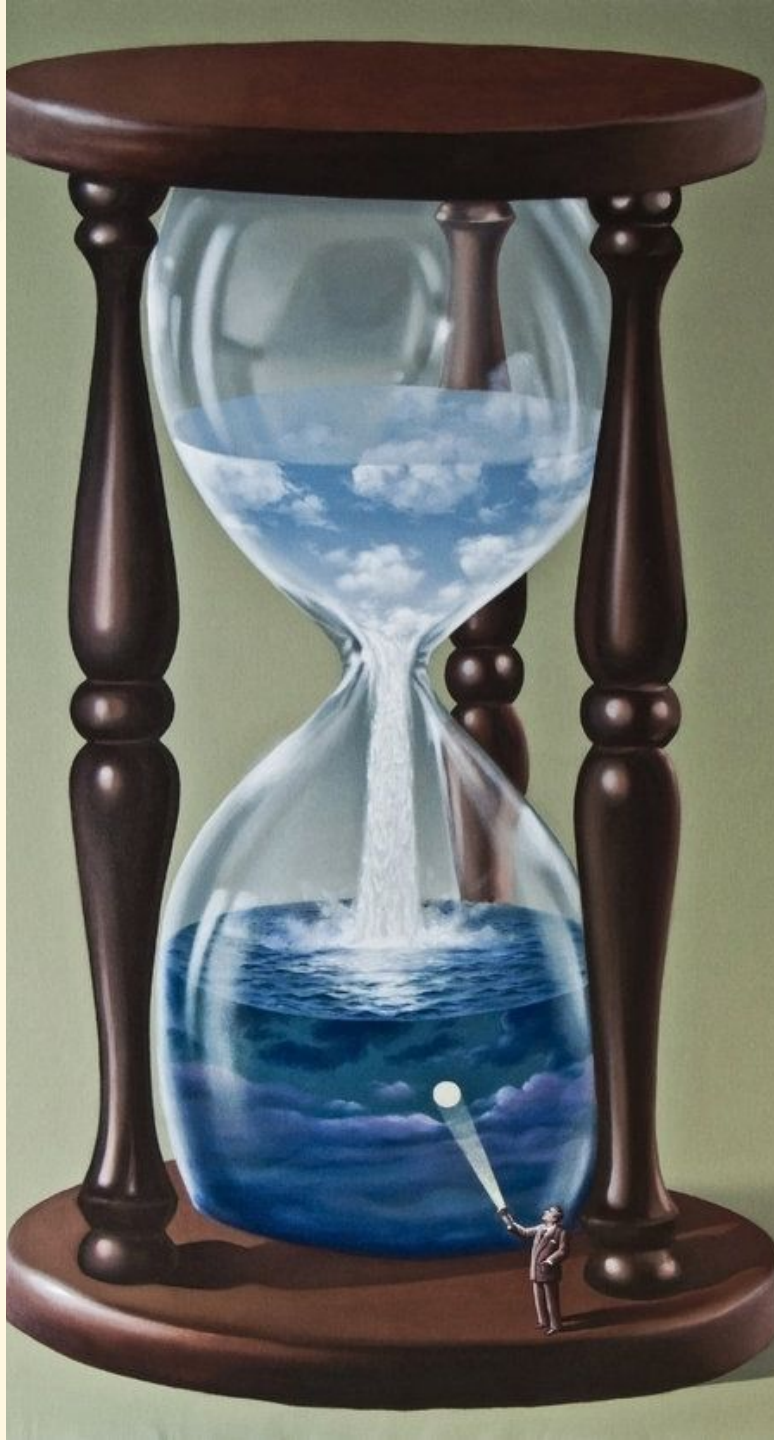
- Inserts into Vector DB
- Vector Embedding Compute
- Inserts into Postgres

@cdxker - denzell.f@trieve.ai

Vector Embedding Compute Latency

- Cloud API's take ~150ms, but 15ms possible w/ Candle
- Ingesting 1,000,000 chunks (33k batches)
  - ~80mins using OpenAI
  - ~8mins on Candle
- LTR plugins usually < 10ms

docs.trieve.ai/vector-inference



Latency by layer

1. Inference vectors for query ~15ms
2. Typo detection+correction ~1ms
3. Score docs in search index ~10ms
4. Re-rank (LTR/cross-encode) ~15ms
5. First LLM token (TTFT) ~250ms

@cdxker - denzell.f@trieve.ai

Explainable Search Results

- Semantic Search doesn't care about keywords. (highlights needed)
- Semantic Search is bad at typos.

Lets look at highlights on hackernews with an example query: "Tools to Track spans"

@cdxker - denzell.f@trieve.ai

The impact of daily journals on personal productivity | Innovations in AI-driven tools for performance analysis

Launch HN: Traceloop (YC W23) – Detecting LLM Hallucinations with OpenTelemetry
101 points by GalKlm 4 months ago | 17 comments | Score 0.531 | **Get Recommendations** | **Add to AI Context**

Hey everyone, we are Nir and Gal from Traceloop (https://www.traceloop.com). We help teams understand when their LLM apps are failing or hallucinating at scale. See a demo: https://www.traceloop.com/video or try it yourself at https://www.traceloop.com/docs/demo.

When moving your LLM app to production, significant scale makes it harder for engineers and data scientists alike to understand when their LLM is hallucinating or returning malformed responses. When you get to millions of calls to OpenAI a month, methods like "LLM as a judge" can't work at a reasonable cost or latency. So, what most people we talked to usually do is sample some generations by hand, maybe for some specific important customers, and manually look for errors or hallucinations.

Traceloop is a monitoring platform that detects when your LLM app fails. Under the hood, we built real-time versions of known metrics like faithfulness, relevancy, redundancy, and many others. These are loosely based on some well-known NLP metrics that work well for LLM-generated texts. We correlate them with changes we detect in your system - like updates to prompts or to the model you're using - to detect regressions automatically.

Here are some cool examples we've seen with our customers -

1. Applying our QA relevancy metric to an entity extraction task, we managed to discover issues where the model was not extracting the right entities (like an address instead of a person's name); or returning random answers like "I'm here! What can I help you with today?".

2. Our soft-faithfulness metric was able to detect cases in summarization tasks where a model was completely making up stuff that never appeared in the original text.

One of the challenges we faced was figuring out how to collect the data that we need from our customers' LLM apps. That's where OpenTelemetry came in handy. We built OpenLLMetry (https://github.com/traceloop/openllmetry), and announced it here almost a year ago. It standardized the use of OpenTelemetry to observe LLM apps. We realized that the concepts of traces, spans, metrics, and logs that were standardized with OpenTelemetry can easily extend to gen AI. We partnered with 20+ observability platforms to make sure that OpenLLMetry becomes the standard for GenAI observability and that the data that we collect can be sent to other platforms as well.

We plan to extend the metrics we provide to support agents that use tools, vision models, and other amazing developments in our fast-paced industry.

We invite you to give Traceloop a spin and are eager for your feedback! How do you track and debug hallucinations? How much has that been an issue for you? What types of hallucinations have you encountered?

Techniques and Tools for Tracking Productivity (lifebyexperimentation.com)
1 points by ZaneClaes December 9, 2014 | context | Score 0.731 | **Get Recommendations** | **Add to AI Context**

Quick and easy tracking (blog.forecast.it)

Without Highlights

@cdxker - denzell.f@trieve.ai

The impact of daily journals on personal productivity | Innovations in AI-driven tools for performance analysis

Launch HN: Traceloop (YC W23) – Detecting LLM Hallucinations with OpenTelemetry
101 points by GalKlm 4 months ago | 17 comments | Score 0.531 | **Get Recommendations** | **Add to AI Context**

Hey everyone, we are Nir and Gal from Traceloop (https://www.traceloop.com). We help teams understand when their LLM apps are failing or hallucinating at scale. See a demo: https://www.traceloop.com/video or try it yourself at https://www.traceloop.com/docs/demo.

When moving your LLM app to production, significant scale makes it harder for engineers and data scientists alike to understand when their LLM is hallucinating or returning malformed responses. When you get to millions of calls to OpenAI a month, methods like “LLM as a judge” can’t work at a reasonable cost or latency. So, what most people we talked to usually do is sample some generations by hand, maybe for some specific important customers, and manually look for errors or hallucinations.

Traceloop is a monitoring platform that detects when your LLM app fails. Under the hood, we built real-time versions of known metrics like faithfulness, relevancy, redundancy, and many others. These are loosely based on some well-known NLP metrics that work well for LLM-generated texts. We correlate them with changes we detect in your system - like updates to prompts or to the model you’re using - to detect regressions automatically.

Here are some cool examples we’ve seen with our customers -

1. Applying our QA relevancy metric to an entity extraction task, we managed to discover issues where the model was not extracting the right entities (like an address instead of a person’s name); or returning random answers like “I’m here! What can I help you with today?”.

2. Our soft-faithfulness metric was able to detect cases in summarization tasks where a model was completely making up stuff that never appeared in the original text.

One of the challenges we faced was figuring out how to collect the data that we need from our customers' LLM apps. That’s where OpenTelemetry came in handy. We built OpenLLMetry (https://github.com/traceloop/openllmetry), and announced it here almost a year ago. It standardized the use of OpenTelemetry to observe LLM apps. We realized that the concepts of traces, **spans**, metrics, and logs that were standardized with OpenTelemetry can easily extend to gen AI. We partnered with 20+ observability platforms to make sure that OpenLLMetry becomes the standard for GenAI observability and that the data that we collect can be sent to other platforms as well.

We plan to extend the metrics we provide to support agents that use **tools**, vision models, and other amazing developments in our fast-paced industry.

We invite you to give Traceloop a spin and are eager for your feedback! How do you **track** and debug hallucinations? How much has that been an issue for you? What types of hallucinations have you encountered?

Techniques and **Tools** for **Track**ing Productivity (lifebyexperimentation.com)
1 points by ZaneClaes December 9, 2014 | context | Score 0.731 | **Get Recommendations** | **Add to AI Context**

Quick and easy **track**ing (blog.forecast.it)

With Highlights

Semantic Search biases towards larger content

@cdxker - denzell.f@trieve.ai

Fast Typo Correction

- Many vector db's don't correct typos
  - Harder when multi-tenant
- BK-Tree vs. SymSpell Data Structures
  - SymSpell is 100x faster

trieve.ai/building-blazingly-fast-typo-correction-in-rust

Sharding your search index Pt. 1

TLDR: think about it

What is a shard?

- A shard is an instance of the search lib (Lucene, FAISS, etc.)
- Shards are made up of segments
- Segments are indices (IDF or HNSW)
- Small segments should be merged off-peak hours

@cdxker - denzell.f@trieve.ai

Sharding your search index Pt. 2

How to optimize?

- Usually best to think about sharding before HNSW params
- Always have `shards > 2*nodes` such that you can horizontally scale CPU if needed
- Replicate shards for more read throughput
- Different db's map their thread pools to shards differently and you should work with your vendor to address it

@cdxker - denzell.f@trieve.ai

Questions?

Thank you!

@cdxker - denzell.f@trieve.ai