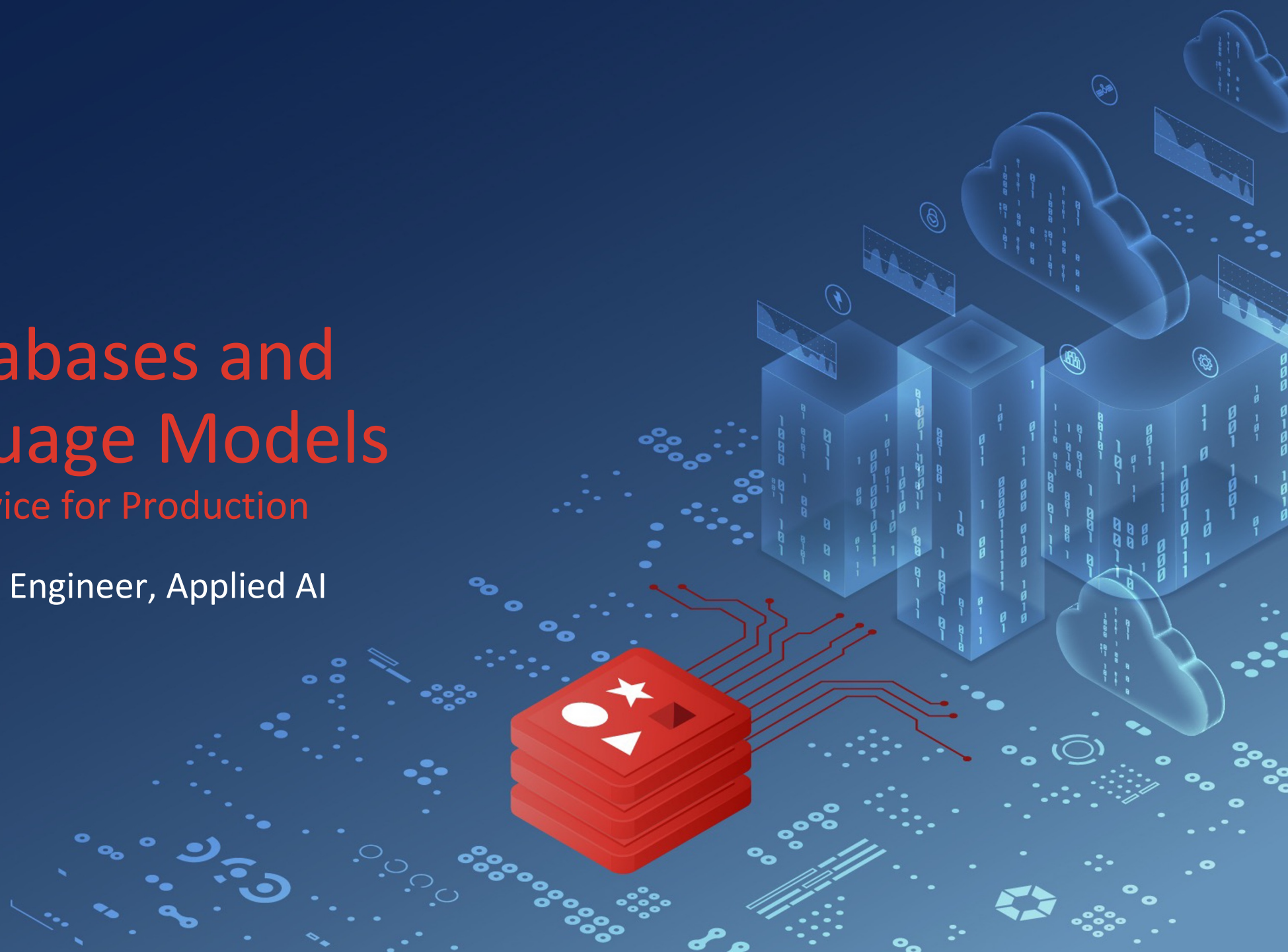
Vector Databases and Large Language Models

Part II - Practical Advice for Production

Sam Partee – Principal Engineer, Applied AI



Vector Embeddings

What are vector embeddings and how are they created?

Vectors

- Commonly represent unstructured data
  - Audio, text, images, etc
- Usually of high-dimension in the form of a **dense** embedding.
- Packed with information
- Easy to use API to create

Vector Embedding Creation

- Simple creation APIs
- Example with HuggingFace Sentence Transformer

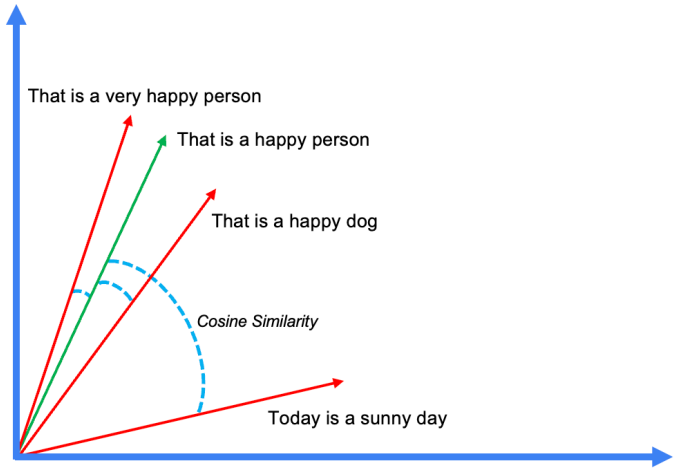
```python
from sentence_transformers import SentenceTransformer
model = SentenceTransformer('sentence-transformers/all-MiniLM-L6-v2')

sentences = [
  "That is a very happy Person",
  "That is a Happy Dog",
  "Today is a sunny day"
]
embeddings = model.encode(sentences)
```

Vector Similarity Search

How are vector embeddings used for similarity search?

- 3 semantic vectors = **Search Space**
  - “today is a sunny day”
  - “that is a very happy person”
  - “that is a very happy dog”
- 1 Semantic vector = **Query**
  - “That is a happy person”

Goal: Find most similar vector to the query



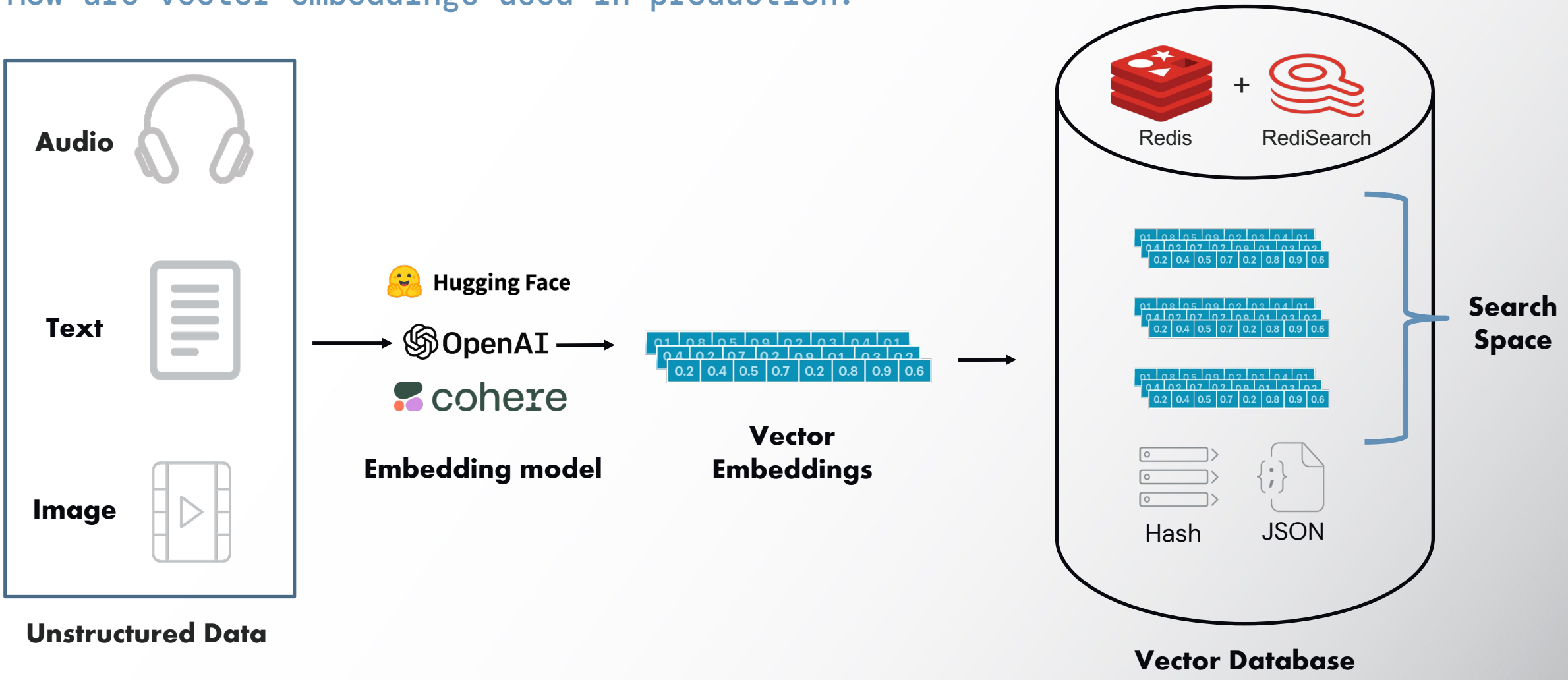
How? Calculate the distance (ex. Cosine Similarity)

That is a happy dog	0.695
That is a very happy person	0.943
Today is a sunny day	0.257

https://mlops.community/vector-similarity-search-from-basics-to-production/

Vector Database

How are vector embeddings used in production?



Redis – Vector Database

Redis + RediSearch

Integrations

OpenAI Retrieval Plugin

LangChain

metal.

Relevance AI

- **Redis**: Low-latency, scalable, in-memory database
- Indexing methods
  - HSNW (ANN)
  - Flat (KNN)
- Distance metrics
  - L2, Cosine, internal product
- Support for hybrid queries
  - Vector search + filtering by text, geo, etc.
- Store vectors in JSON (new in 2.6)

Design Patterns

For using Large Language Models with Vector Databases

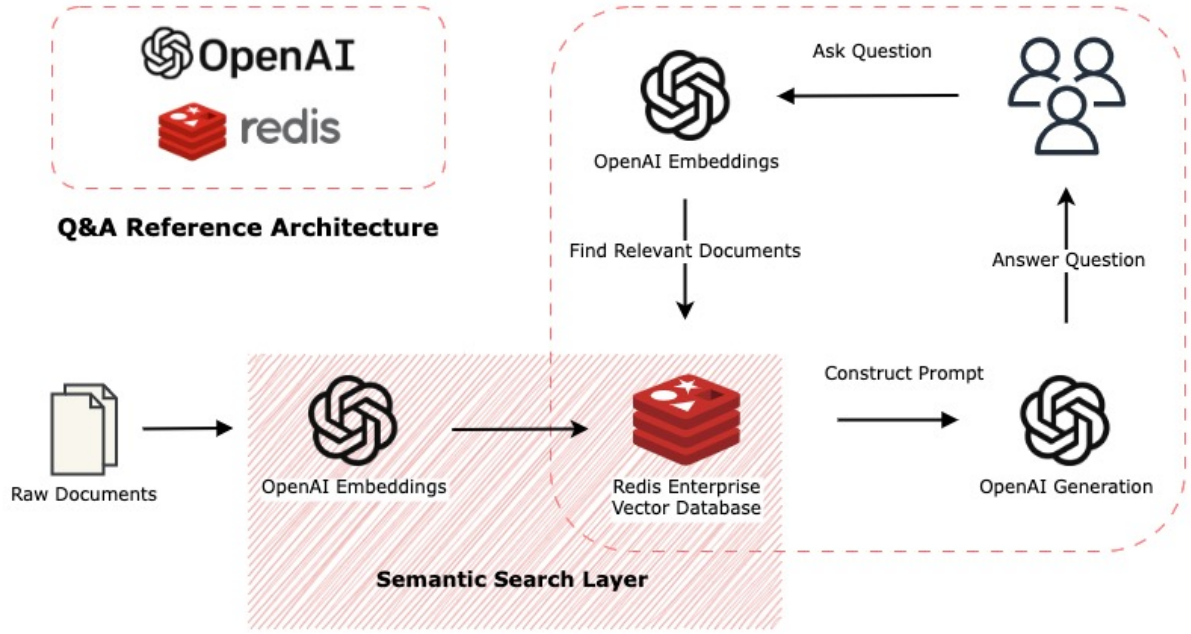
88% believe a retrieval mechanism, such as a vector database, would remain a key part of their stack. Retrieving relevant context for a model to reason about helps increase the quality of results, reduce “hallucinations” (inaccuracies), and solve data freshness issues.

- Sequoia LLM Survey

https://www.sequoiacap.com/article/llm-stack-perspective/

Context Retrieval

Finding relevant information for LLM queries



- Description
  - Vector database is used as an external knowledge base for the large language model.
  - Queries are used to detect similar information (context) within the knowledge base
- Benefits
  - **Cheaper and faster** than fine-tuning
  - **Real-time updates** to knowledge base
  - **Sensitive data** doesn't need to be used in model training or fine tuning
- Use Cases
  - Document discovery and analysis
  - Chatbots

Document QnA Example: https://github.com/RedisVentures/redis-openai-qna

Chatbot Example w/ Langchain: https://github.com/RedisVentures/redis-langchain-chatbot

Context Retrieval

Advanced Search Techniques (HyDE)

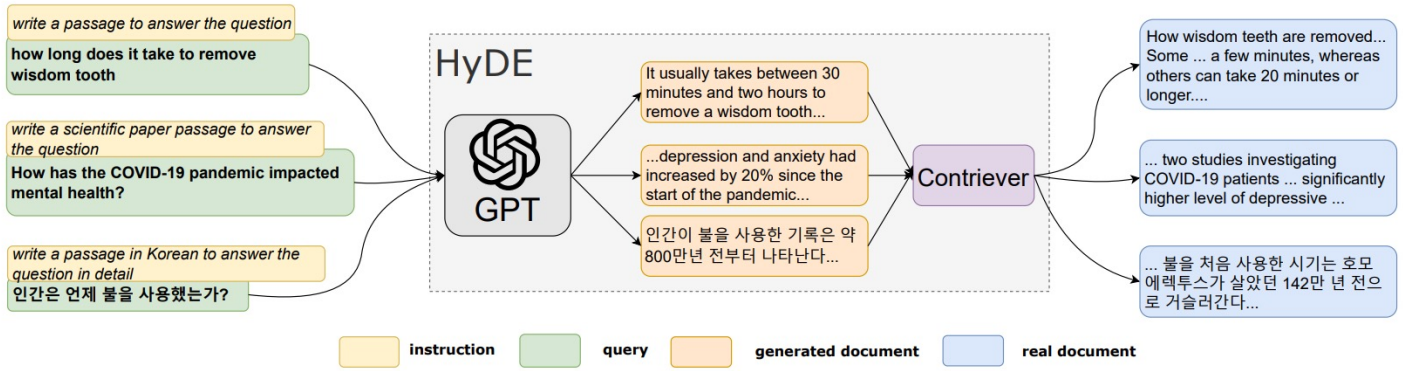


Figure 1: An illustration of the HyDE model. Documents snippets are shown. HyDE serves all types of queries without changing the underlying GPT-3 and Contriever/mContriever models.

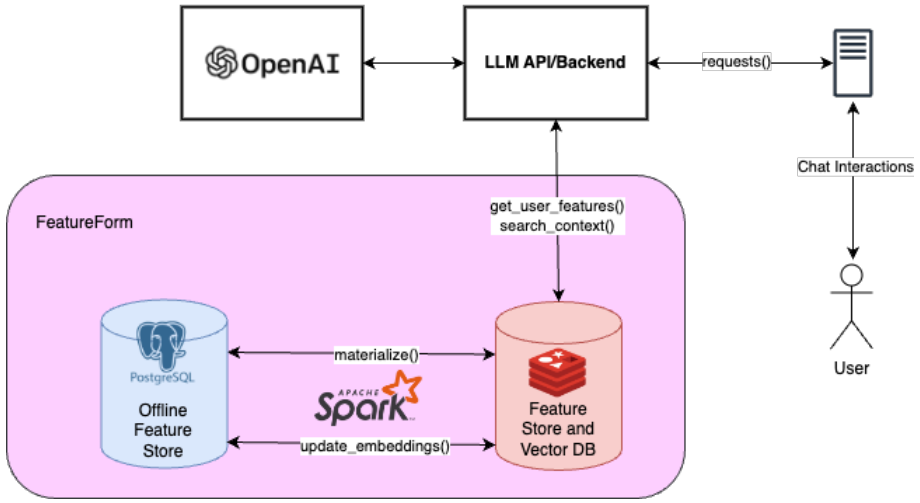
Precise Zero-Shot Dense Retrieval without Relevance Labels

https://arxiv.org/pdf/2212.10496.pdf

- Description
  - Use summarization to improve the prompt of a question-and-answer loop.
- Benefits
  - **Improves** answer quality and correctness
- Drawbacks
  - **Slow.**
  - Requires multiple trips to LLM model which may take seconds...
- How to use
  - As a backup plan to QnA when answer isn't found or when threshold for confidence isn't met.
  - Run concurrently to normal loop (expensive)

Feature Injection

Providing real-time features to prompts

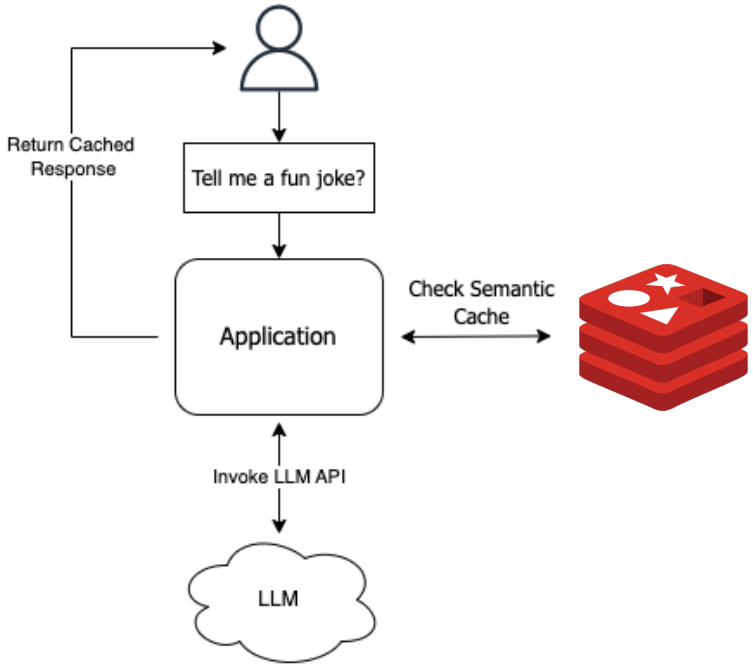


https://blog.langchain.dev/feature-stores-and-llms/

- Description
  - Features for catalogue (product, user, etc) available for retrieval in prompt.
  - Use traditional feature orchestration for LLM workflow to augment prompt
  - Prompt contains entity-specific information
- Benefits
  - Allows for **real-time inclusion of entity information** (user/product/etc)
  - **Improves** user experience by tailoring interactions.
  - **Secure.** No other user gets another users info
- Feature Orchestration Providers
  - FeatureForm (both VectorOps and Feature)
  - Tecton

Semantic Caching

Reducing cost and improving QPS of LLMs



- Description
  - Vector database used to cache similar queries and answers
  - Queries embedded and used as a cache lookup prior to LLM invocation
- Benefits
  - **Saves on computational and monetary cost** of calling LLM models.
  - Can **speed up applications** (LLMs are slow)
- Use Cases
  - Every single use case we've talked about that uses an LLM.

LLM Guardrails

Controlling the output of an LLM Model

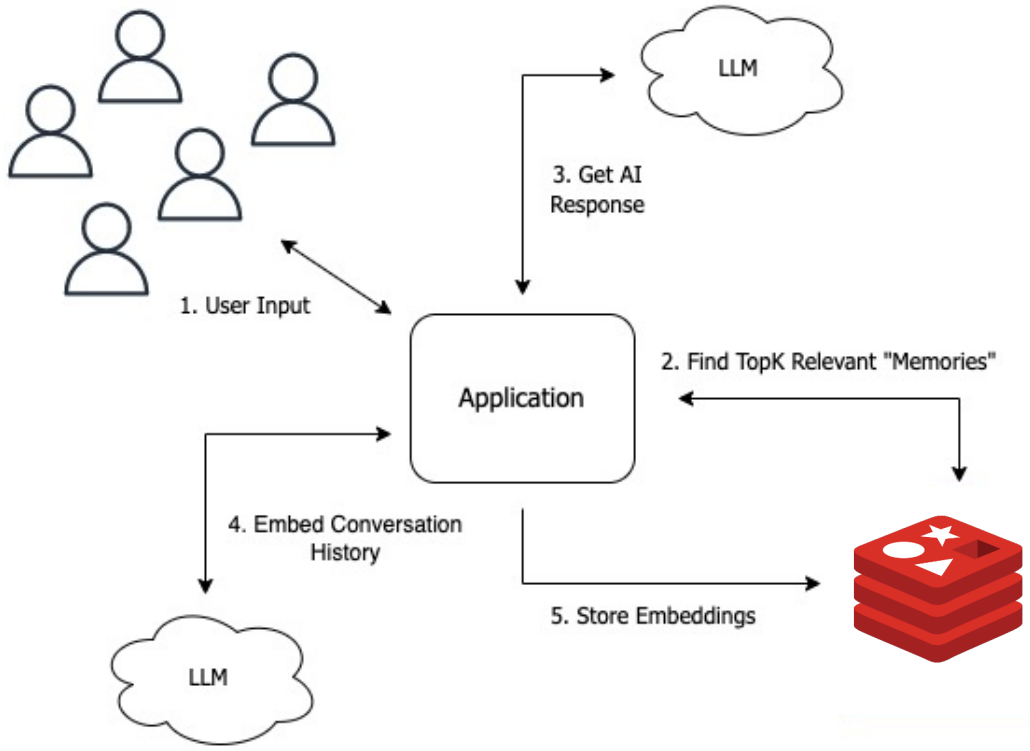
```
define user express greeting
  "Hello!"
  "Good afternoon!"

define flow
  user express greeting
  bot express greeting
  bot offer to help

define bot express greeting
  "Hello there!"

define bot offer to help
  "How can I help you today?"
```

https://github.com/NVIDIA/NeMo-Guardrails

- Description
  - LLMs are prone to hallucinations.
  - Vector databases can be used as a method to reduce out-of-context answers
  - Use knowledge base to retrieve what context should be understood.
- Benefits
  - **Stops prompt injection**
  - Allows for **behavior to be defined and relevant.**
- Examples
  - NEMO Guardrails (NVIDIA)

Long-Term Memory

Increasing available context to LLMs



- Description
  - Theoretically infinite, contextual memory that encompasses multiple simultaneous sessions
  - Retrieves only last K messages relevant to the current message in the entire history.
- Benefits
  - Provides **solution to context length limitations** of large language models
  - Capable of **addressing topic changes** in conversation without context overflow
- Use Cases
  - Chatbots
  - Information retrieval

Repository: https://github.com/continuum-llms/chatgpt-memory

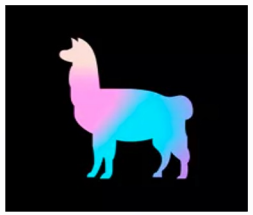
Common Mistakes

for LLM + Vector database usage in production

Common Mistake – Laziness

Context window doesn't solve everything

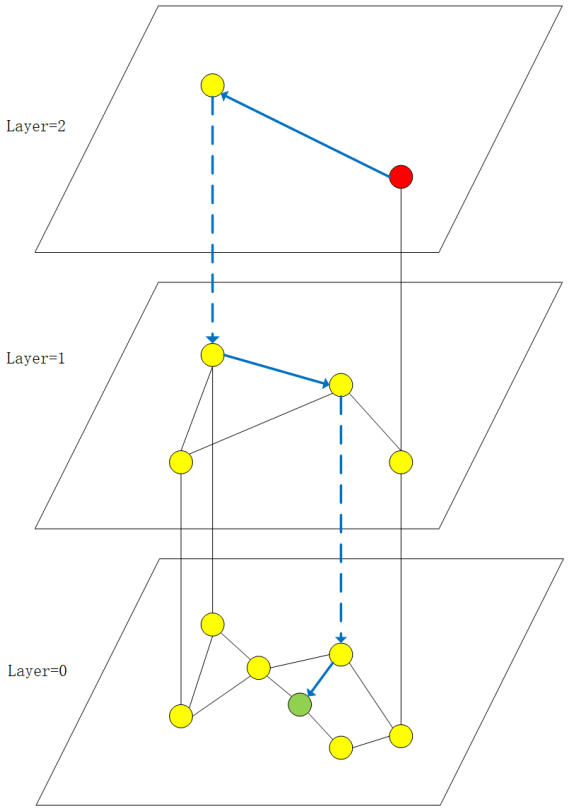
LangChain



- Problem
  - Each use case demands a different combination of context, embeddings, models, and metadata
  - Each integration framework is different
- Factors
  - Context window size?
  - # of retrieved pieces of context?
  - # of tokens per embedding? (chunks)
- Mistake
  - Abusing large context windows.
  - Trusting defaults – Every use case is different
- Solution
  - K-fold test your dataset with above variables
  - Understand your integration framework

Common Mistake – Index Management

Tips for different index scenarios

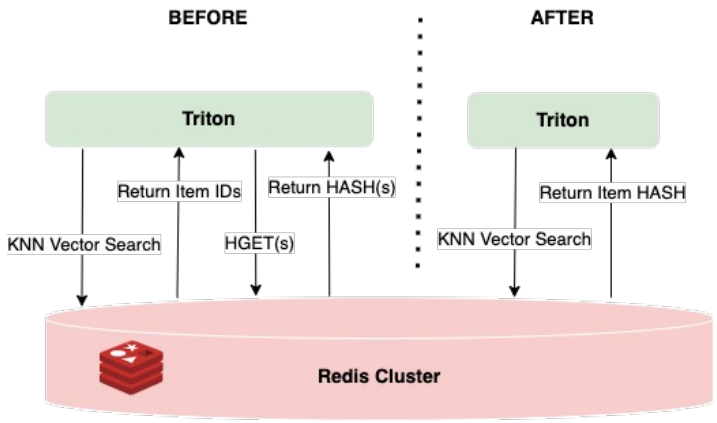


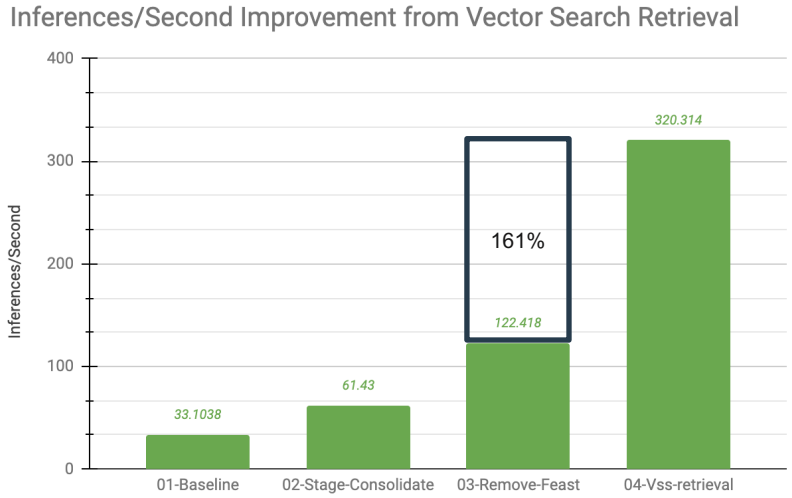
https://github.com/vearch/vearch/wiki/Hnsw-Real-time-Index-Detailed-Design

- Problem
  - How do I setup an index if I have multiple use cases?
- Factors
  - Does vector database charge $ per index?
  - How many indices?
  - How large are they?
  - Does vector database support hybrid?
- Mistake
  - Not thinking ahead about scaling number of indices, size, or cost.
- Solution
  - Project cost, test performance, and mock scale

Common Mistake – Separate Metadata

Where do I store data besides the vector data?





- Problem
  - Metadata retrieval for prompt or filtering
- Factors
  - Does vector database support hybrid?
  - How many network trips to use metadata?
  - Are metadata and vectors co-located?
  - Desired QPS/latency?
- Mistake
  - Using platforms or client implementations that increase latency or complexity of infra due to separation of metadata and vectors
- Solution
  - Profile network calls.
  - Organize schema efficiently

https://github.com/RedisVentures/Redis-Recsys

https://www.nvidia.com/en-us/on-demand/session/gtcspring23-s51222/

Architecture Considerations

for LLM + Vector database usage in production

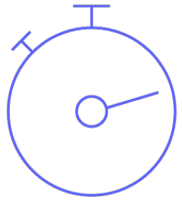
Considerations for LLMs in Production

Cost

- Costly hardware requirements to self host
- Repetitive API calls for embedding creation, content generation, or model fine tuning

Reliability

- Model "*hallucinations*"
- Lack of clarity in AI model biases
- What about domain-specific content?



Latency

- LLMs are just too slow for modern, real-time applications
- Unpredictable traffic load and rate limits on public services

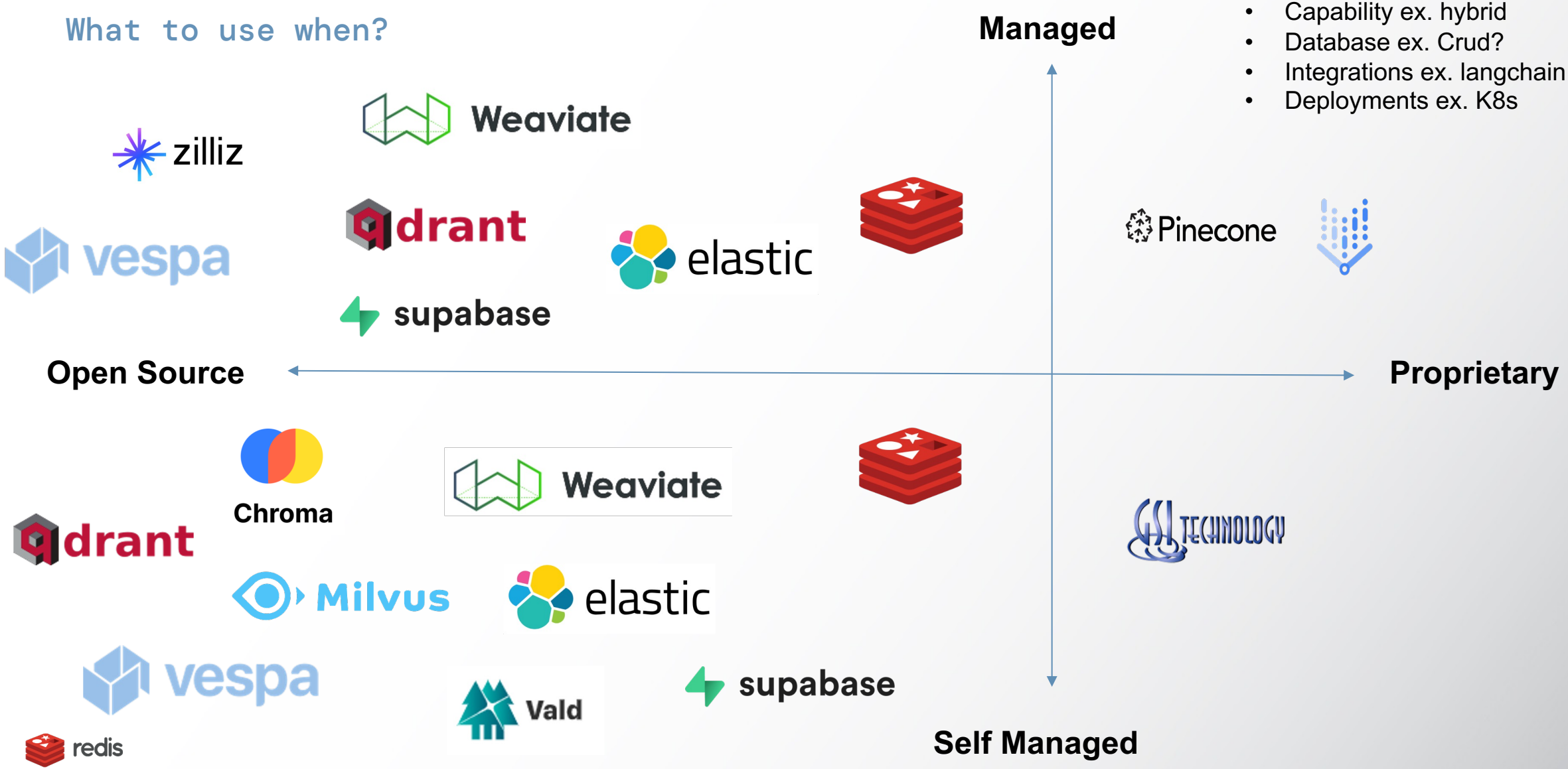
Security

- Publicly used APIs & shareable keys
- Proprietary data likely can't be used in model training or fine tuning
- Model quality has security implications

Vector Algo? Vector Database?

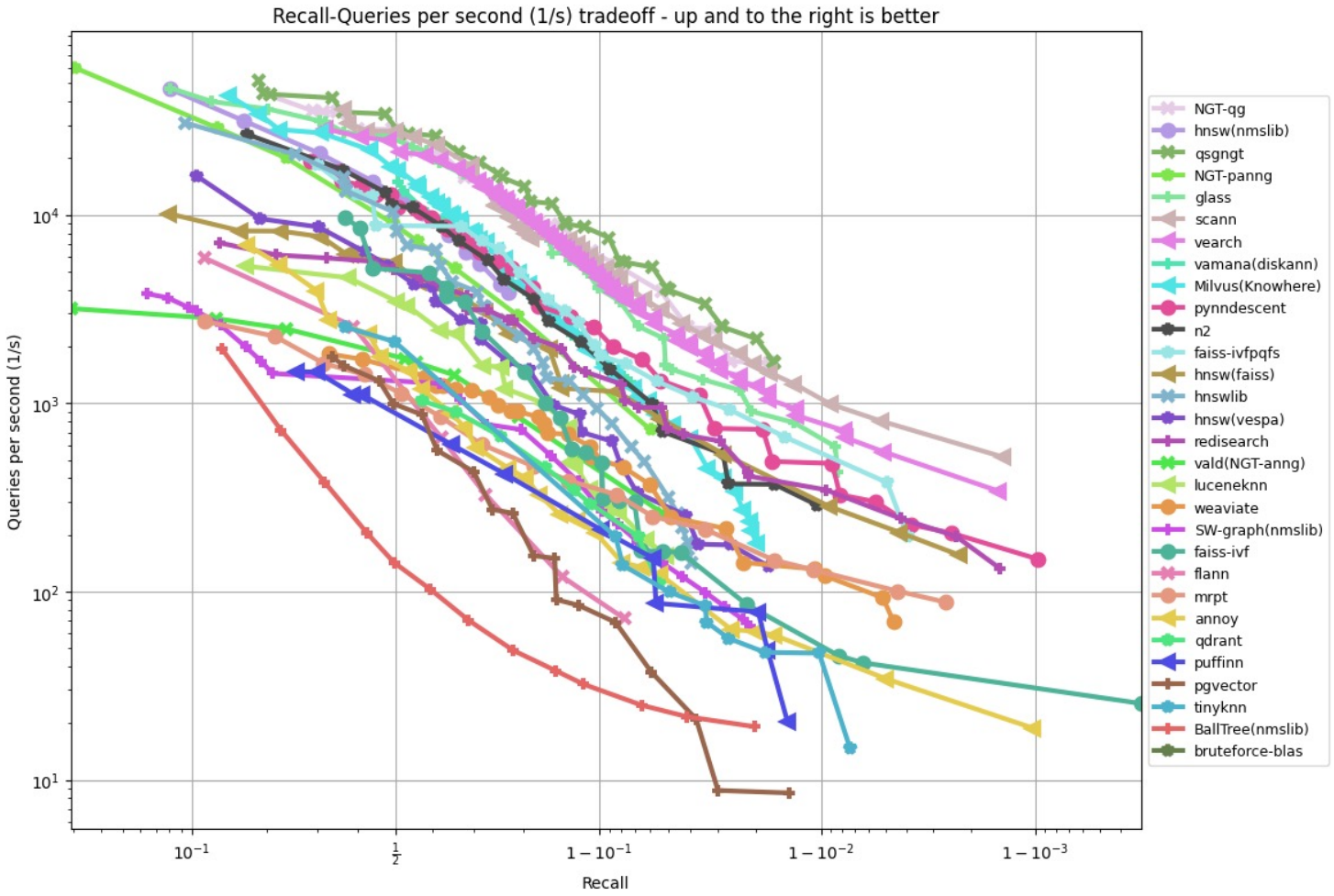
What to use when?

Considerations
- Team size, ability, budget
- Performance
- Capability ex. hybrid
- Database ex. Crud?
- Integrations ex. langchain
- Deployments ex. K8s



Vector Database Performance

Queries per Second (QPS) to Recall

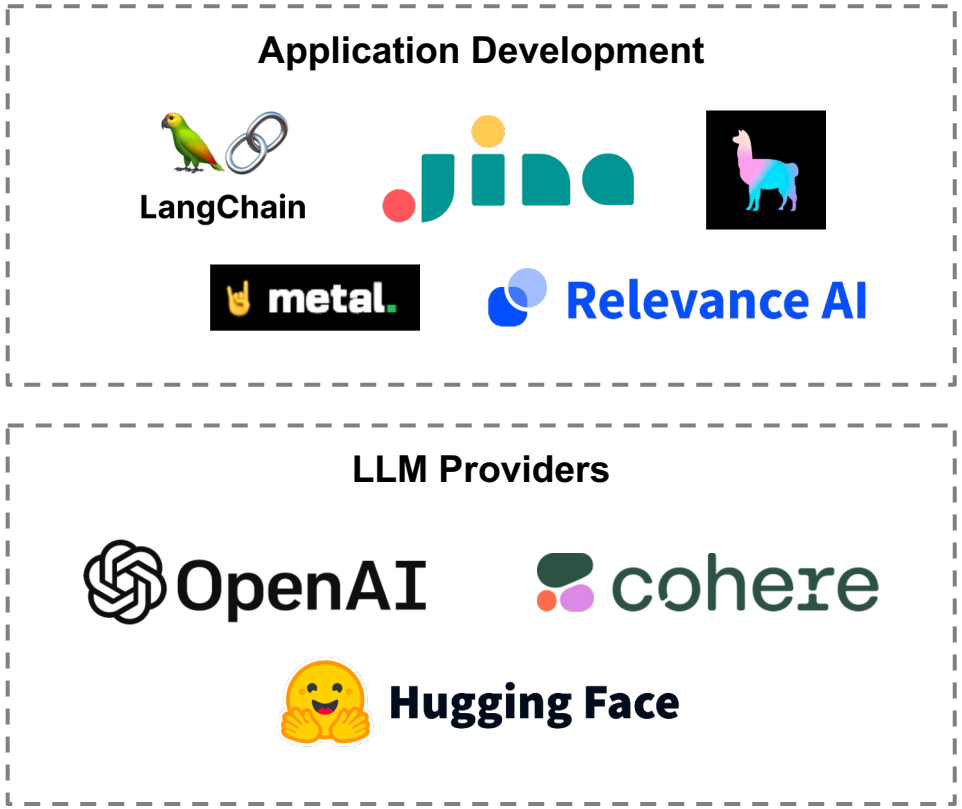


- Performance for ANN benchmarks commonly calculated by QPS/recall
- Importance of this varies by use case
  - Rec-Sys? **Important**
  - Chatbots? Less Important
- Higher in performance usually leads to having less features
  - PQ for scale, but then maintain cookbook
  - Faiss re-training
  - Hybrid search
  - Database features

https://github.com/erikbern/ann-benchmarks

Providers

Embedding, integration, model providers



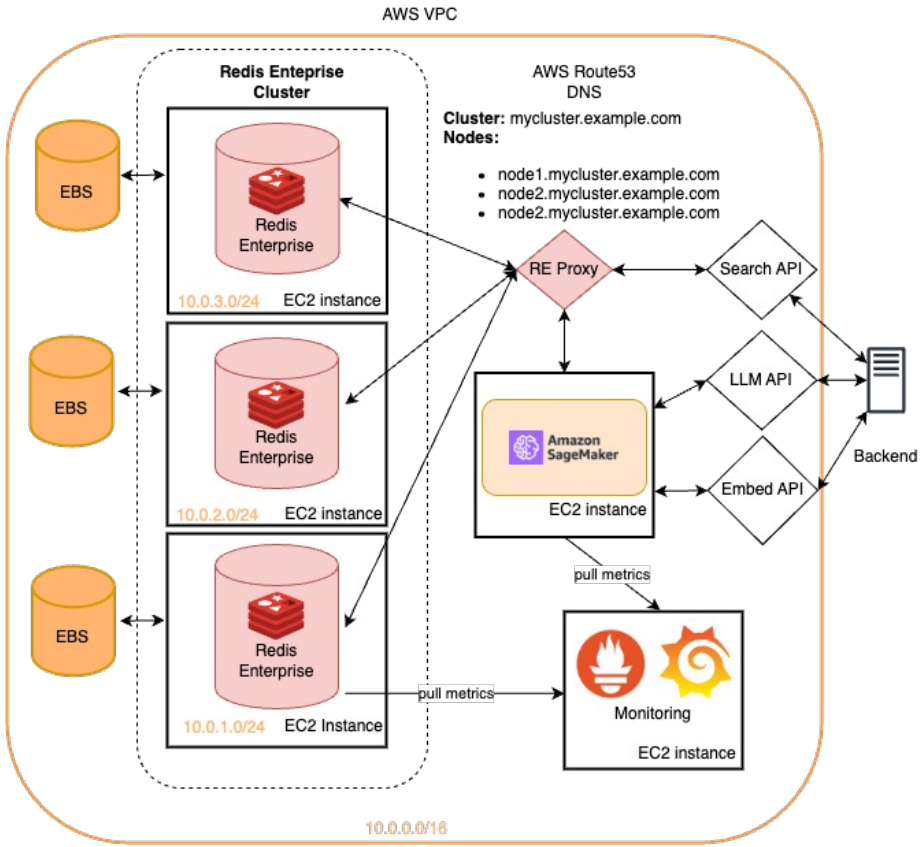
- Questions
  - Open Source vs Closed Source
  - Platform vs roll your own
  - Enterprise readiness
  - Cost
  - Scale and performance
- Integration providers
  - Ex. Langchain vs Relevance
- Model Providers
  - Ex. Huggingface vs OpenAI or Cohere
  - Infra cost vs model cost? Dev time?
- Embeddings Providers
  - Model/embedding manipulation? Fine-tuning?

Example Architectures

for LLM + Vector database usage in production

AWS-based Question and Answer

Example Architecture

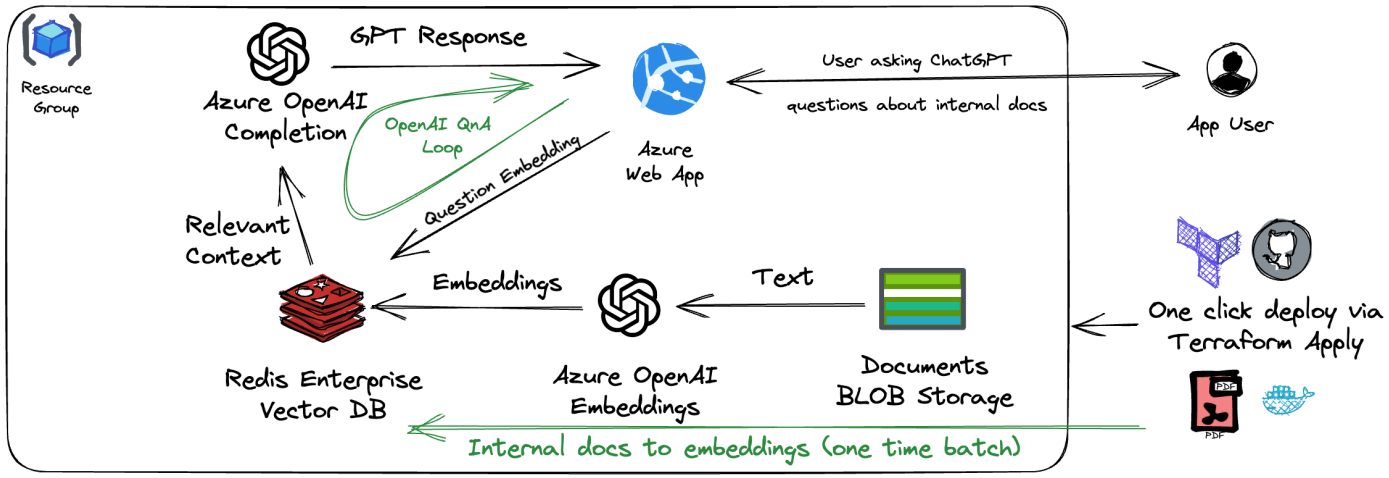


- Infrastructure
  - **Vector Database:** Redis Enterprise
  - **Inference:** Sagemaker
  - **LLM:** Cohere Multilingual
  - **Monitoring:** Prometheus/Graphana
  - **API layer:** FastAPI
  - **CSP:** AWS
  - **Deployment:** Terraform
- Use Cases
  - QnA
  - Search
  - Chatbot

https://github.com/Redislabs-Solution-Architects/tfmodule-aws-redis-enterprise-deploy-cluster

Azure-based Document Intelligence

Example Architecture

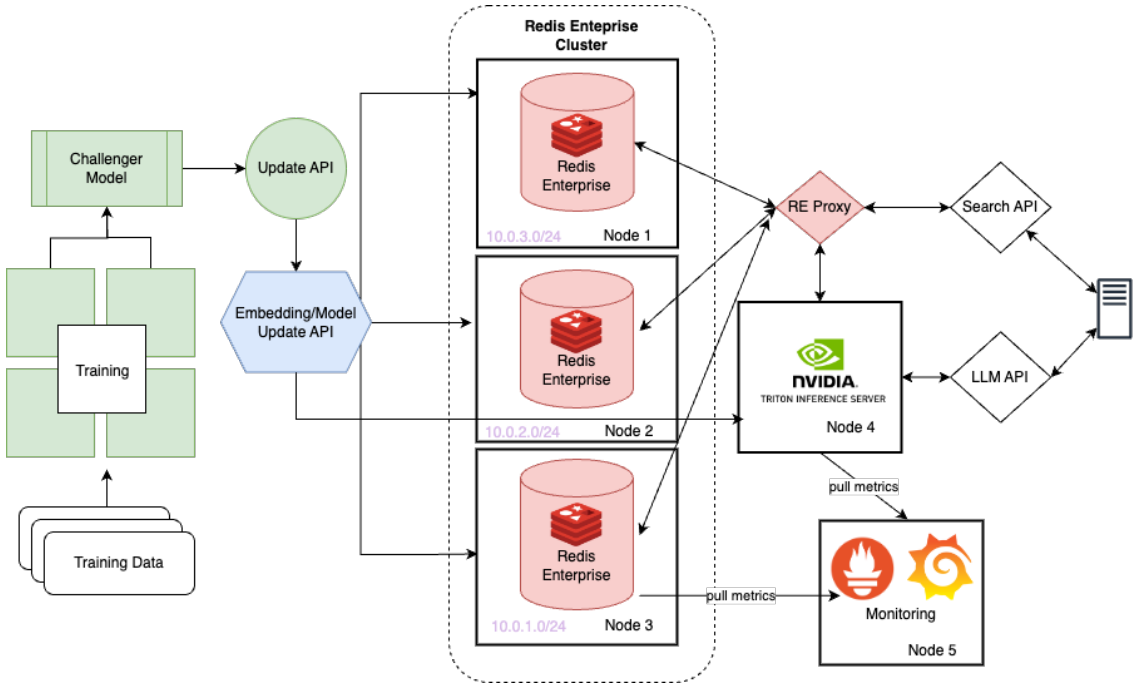


- Infrastructure
  - **Vector Database:** Redis Enterprise (ACRE)
  - **Inference:** Azure ML Studio
  - **LLM:** Azure OpenAI
  - **Monitoring:** RedisInsight
  - **API layer:** FastAPI
  - **CSP:** Azure
  - **Deployment:** Terraform
- Use Cases
  - Document Intelligence
- Benefits
  - Compliance

Terraform: https://github.com/RedisVentures/azure-openai-redis-deployment
Codebase: https://github.com/ruoccofabrizio/azure-open-ai-embeddings-qna

General on-premise LLM + VDB

Example Architecture



- Infrastructure
  - **Vector Database:** Redis Enterprise (ACRE)
  - **Inference:** NVIDIA Triton
  - **LLM:** Custom
  - **Monitoring:** Prometheus/Graphana
  - **API layer:** FastAPI
  - **CSP:** None
  - **Deployment:** Custom
- Use Cases
  - All of the above
- Benefits
  - Cost scaling
  - Deployment and fine-tuning flexibility

How to get started?

`Where do I go next?`

- Getting Started Resources
  - OpenAI Cookbook: https://github.com/openai/openai-cookbook/tree/main/examples/vector_databases/redis
  - Redis Ventures: https://github.com/redisventures
- Examples
  - **LLM Document Chat:** https://github.com/RedisVentures/LLM-Document-Chat
  - **OpenAI QnA:** https://github.com/RedisVentures/redis-openai-qna
- More information
  - Follow me @sampartee or add me on linkedin
  - For slides and talk, check partee.io in a few days.