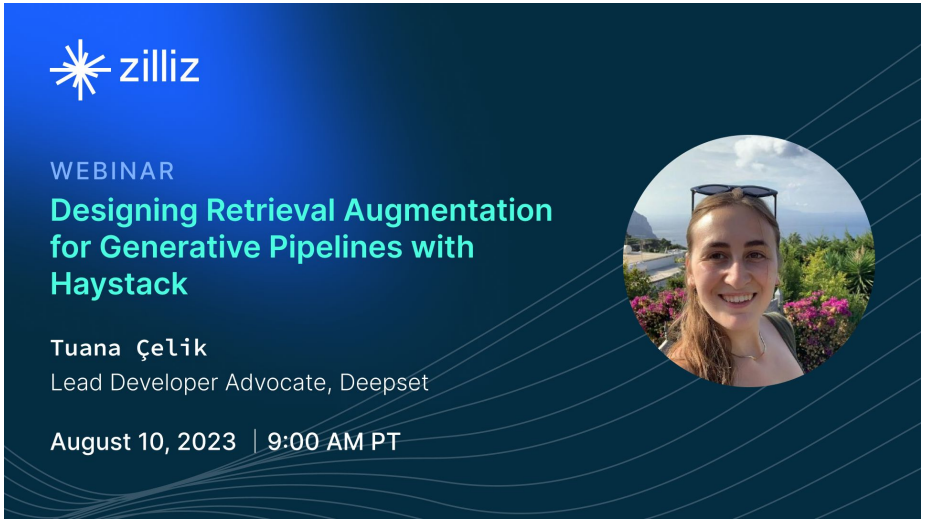
Generative AI and Ruby

Andrei Bondarev, Yujian Tang

Upcoming Events



zilliz.com/event

Generative AI and Ruby

Andrei Bondarev, Yujian Tang

Speaker

Andrei Bondarev

Solutions Architect,
Source Labs LLC

andrei@sourcelabs.io
https://www.linkedin.com/in/andreibondarev/
https://twitter.com/rushing_andrei

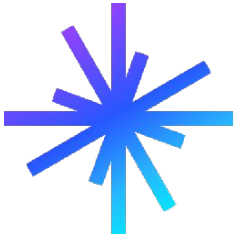
Speaker

Yujian Tang

Developer Advocate, Zilliz

yujian@zilliz.com
https://www.linkedin.com/in/yujiantang
https://www.twitter.com/yujian_tang

Company



zilliz

@Zilliz_Universe

linkedin.com/in/zilliz

milvusio.slack.com

github.com/milvus-io/milvus

zilliz.com

CONTENTS

01 Introduction to Generative AI

02 Generative AI with Ruby

03 Milvus with Ruby

04 Langchain.rb (“Langchain for Ruby”)

01

Introduction to Generative AI

Unstructured Data is Everywhere

Unstructured data is any data that does not conform to a predefined data model.

By 2025, IDC estimates there will be 175 zettabytes of data globally (that's 175 with 21 zeros), with **80% of that data being unstructured**. Currently, 90% of unstructured data is never analyzed.

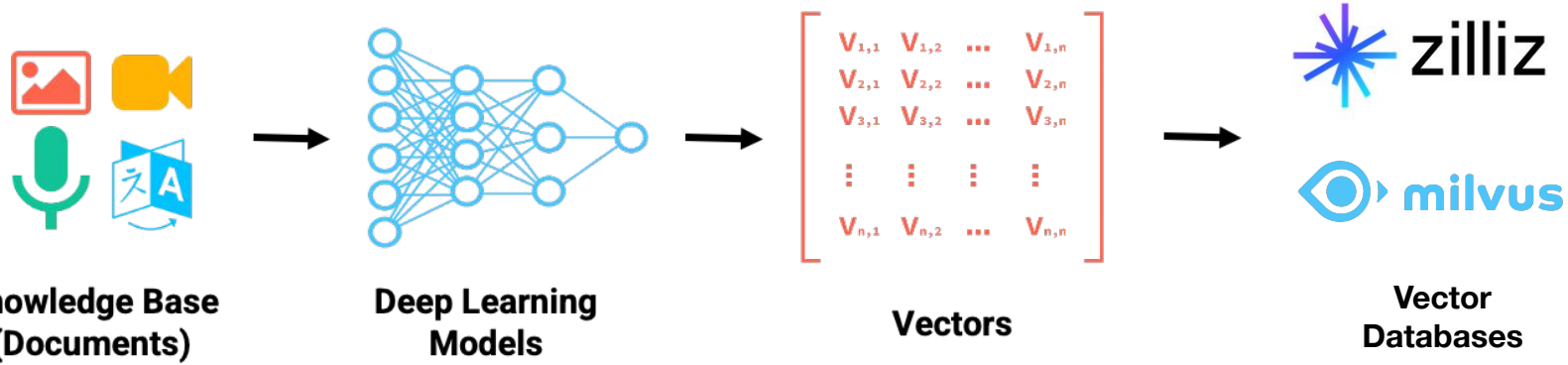
Text

Images

Videos

and more!

How Do You Deal with Unstructured Data?



Vector Embeddings

- Access to Domain Knowledge
- Semantic Search on Domain Knowledge via Vector Embeddings

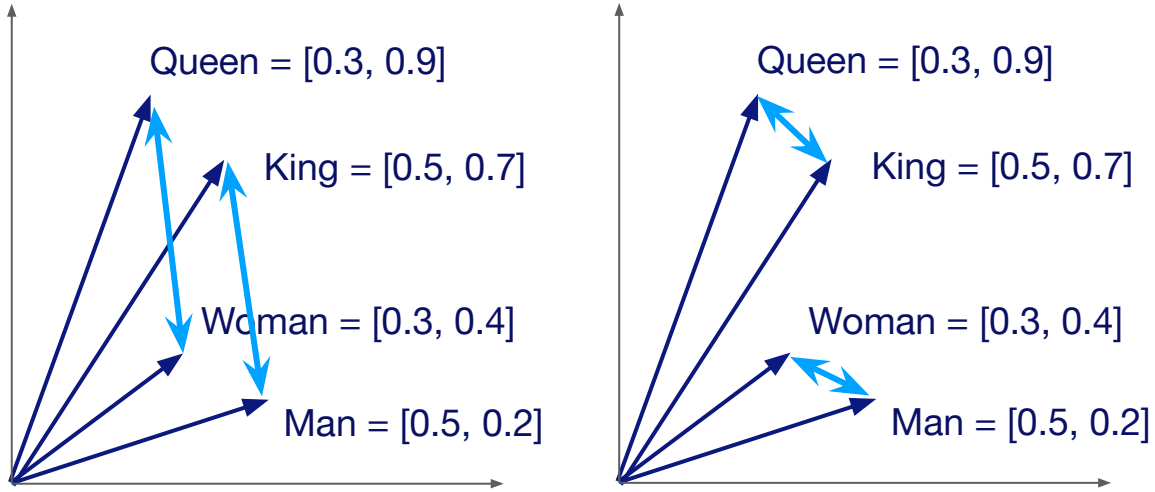
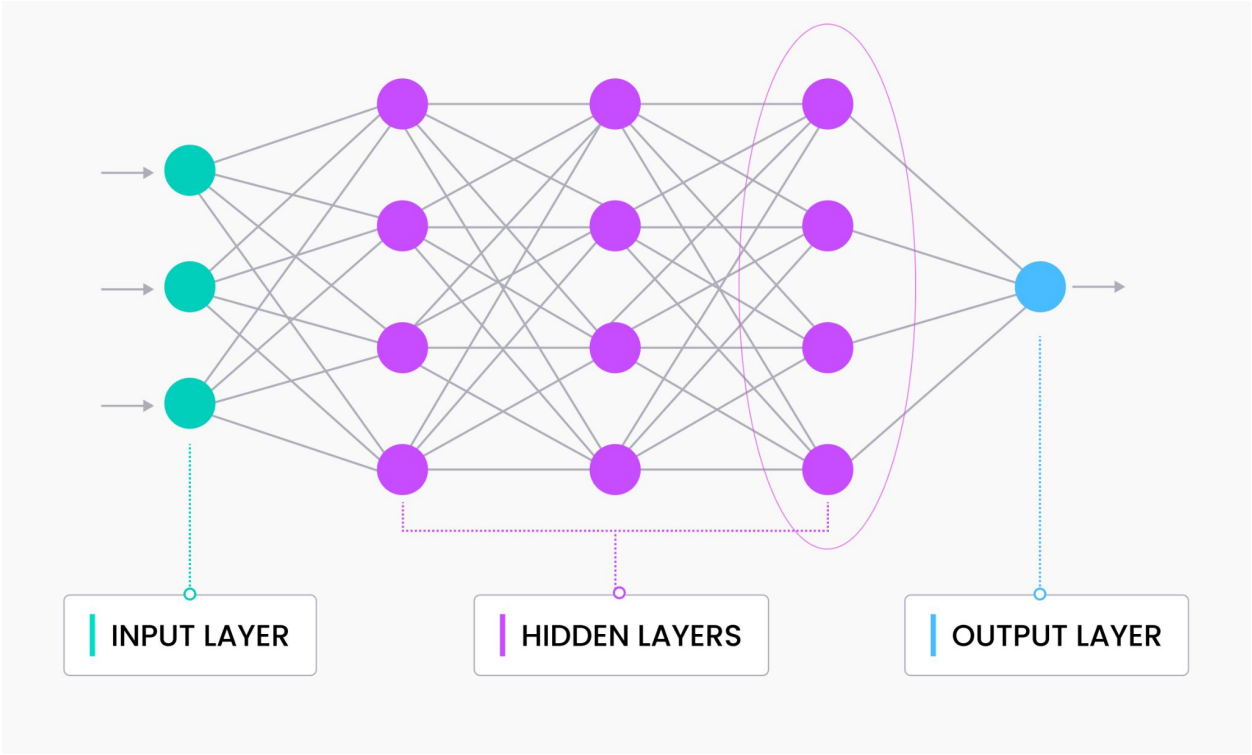


Image from Sutor et al

How are these generated?



Vector Embedding

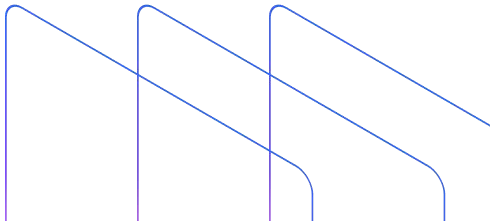
Default project › asdf › medium_articles

medium_articles LOADED

Connection Guide | Actions

Collection Details | Schema | Data Preview | Vector Search

id	title_vector	title	link
125	[0.014838364, -0.017620698, 0.03955149...	The dawn of Dark Mode	https://uxde
195	[-0.008266705, -0.009836753,...	Scrollsuming – The New Zombie Interaction	https://medi
1464	[0.036901046, 0.00553058, 0.011379256,...	Make Issue Types Great Again	https://medi
1538	[0.013065741, 0.011942832, -0.00739964...	Creating an exam archive system with a Data...	https://towar
1623	[-0.0024157332, -0.007975485,...	Using a Golang pattern to write better...	https://medi
2488	[0.067513265, 0.015994877,...	Make Your Data Models Into Websites	https://towar
2926	[-0.009442576, 0.0028793914,...	Create A Machine Learning Model With Goog...	https://towar
3480	[0.048443407, -0.013659755,...	What If Only Batch Normalization Layers Wer...	https://towar
3573	[0.02339675, -0.0032612802, 0.01034519...	What Zynn's entrance mean for the North...	https://medi
4218	[0.054374292, 0.011657661, 0.04338219,...	Vibing Out TensorFlow	https://towar



Where can Vector Databases Help?

Image similarity search

Images made searchable and instantaneously return the most similar images from a massive database.

Video similarity search

Converting key frames into vectors and storing the results, lets billions of videos be recommended in near real time.

Audio similarity search

Quickly query massive volumes of audio data such as speech, music, sound effects, and surface similar sounds.

Molecular similarity search

Blazing fast similarity search, substructure search, or superstructure search for a specified molecule.

Text search engine

Help users find the information they are looking for by comparing keywords against a database of texts.

DNA sequence classification

Accurately sort out the classification of a gene in milliseconds by comparing similar DNA sequence.

Question answering system

Interactive digital QA chatbot that automatically answers user questions.

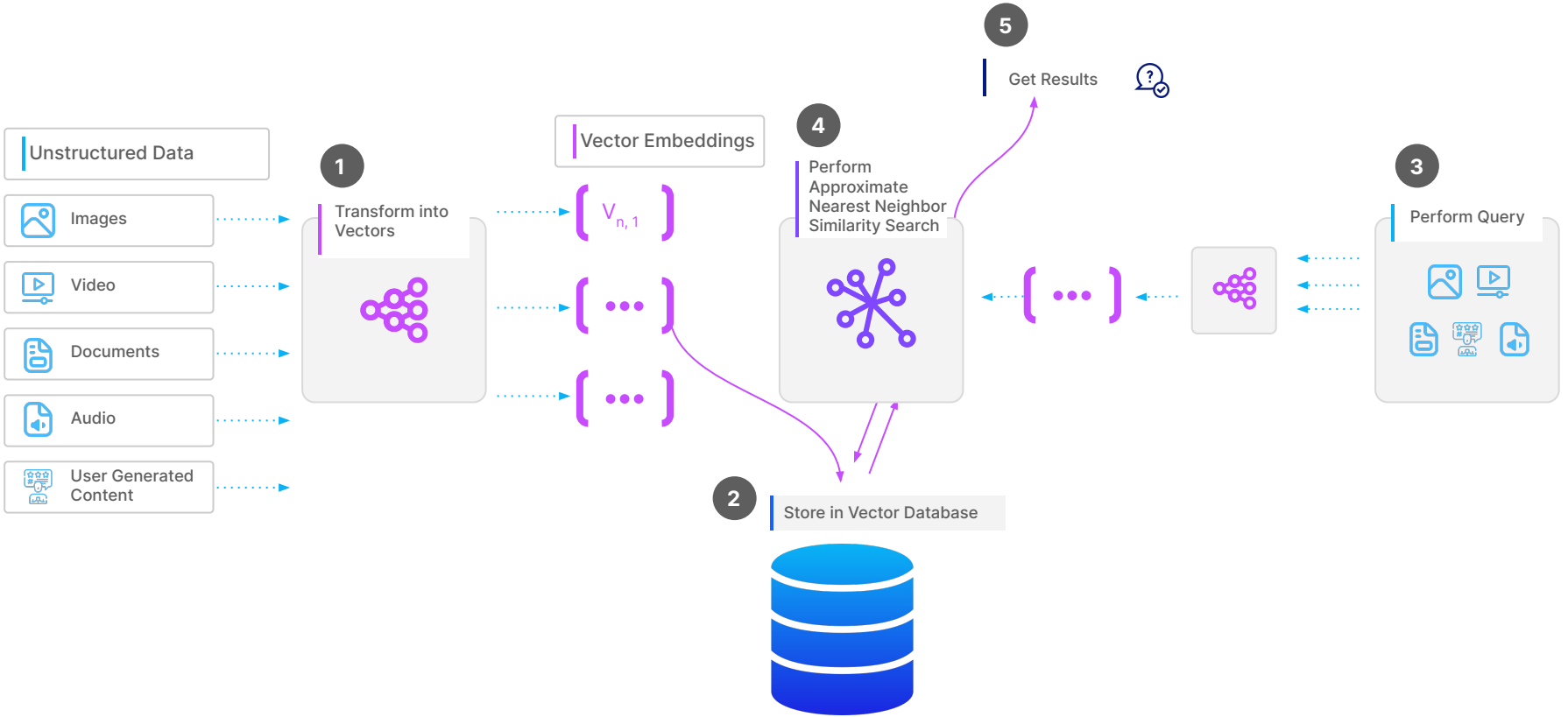
Recommender system

Recommend information or products based on user behaviors and needs.

Anomaly detection

Identifies data points, events, and/or observations that deviate from a dataset's normal behavior.

How Similarity Search Works



Exploring Generative AI Use Cases

Example
A company has 100,000s+ pages of proprietary documentation to enable their staff to service customers.

Problem
Searching can be slow, inefficient, or lack context.

Solution
Create internal chatbot with ChatGPT and a vector database enriched with company documentation to provide direction and support to employees and customers.

02

Generative AI with Ruby

Company

www.sourcelabs.io

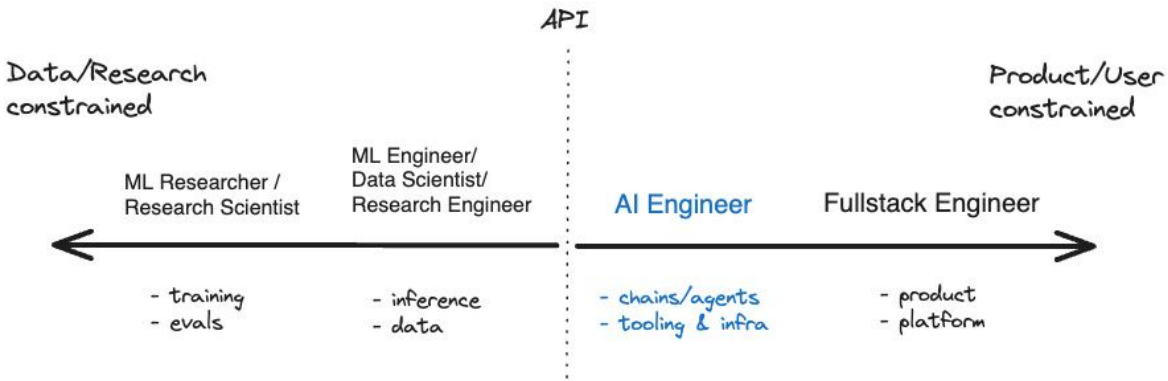
Why Ruby?

Rails

Rapid (pragmatic) application development

“Monolith-first” approach

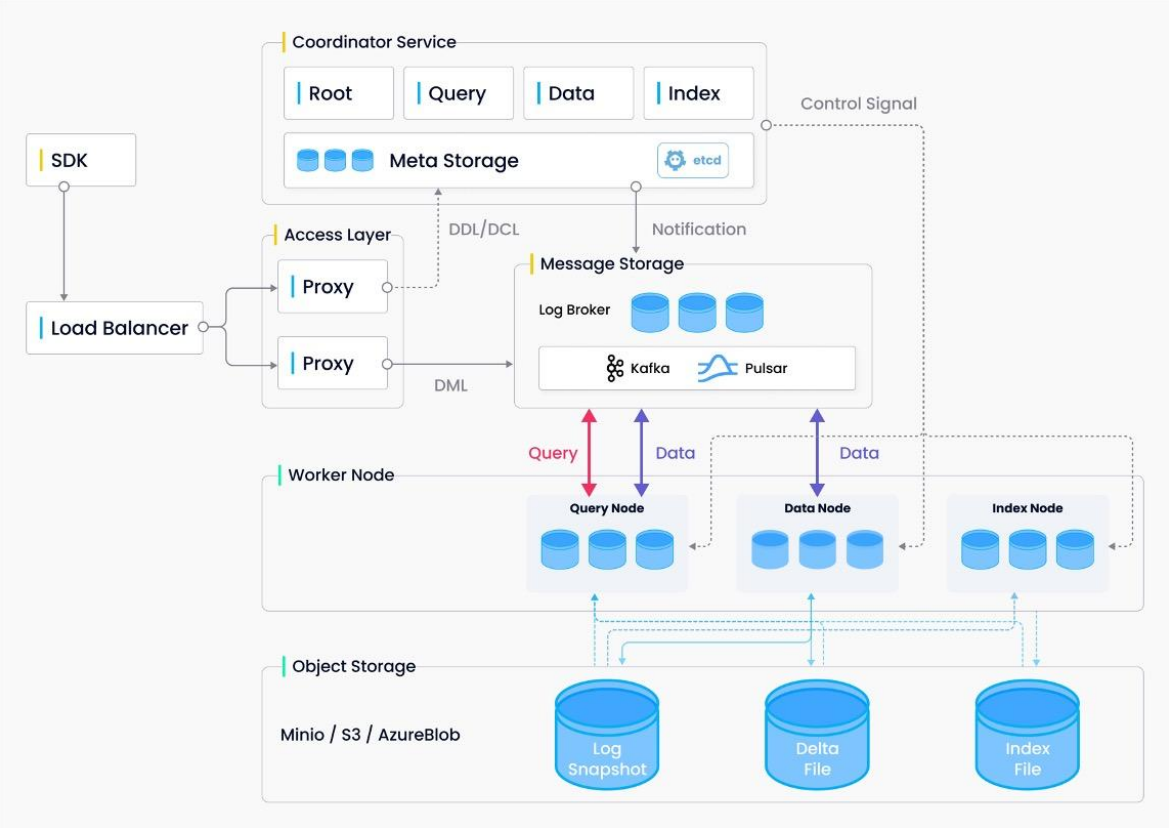
Conducive to the monolith application architecture.



03

Milvus with Ruby

What is Milvus



Milvus API Ruby client

- Github: https://github.com/andreibondarev/milvus
- Popular option for vector search
- Thousands of downloads
- Convenient wrapper in Ruby on top of Milvus
- Part of the Langchain.rb stack

Supports

Creating collections, indices, adding data, searching & querying and managing partitions.

04

Langchain.rb

Langchain.rb

Orchestration Layer
For building LLM-based applications, for stringing multiple types of systems together.

Use-cases
Semantic search/vector search, Q&A over documents, chat bots, and (experimental) agents.

Best practices
Framework encapsulating the best practices for building applications with LLMs

Langchain.rb

Github: github.com/andreibondarev/langchainrb

3 months old

⭐ 480 stars

30+ contributors

10k downloads

Discord community

Features

Prompt Management
Creating, loading and saving prompt templates

Context Length Validation
Validating the context window length to avoid hitting LLM errors

Data Chunking
Splitting the data before indexing into vector search DBs

Conversation Memory
Persisting a chat with an LLM to memory

...

LLMs x Vector Search DBs matrix

	Milvus	Pgvector	Qdrant	Weviate	…
Anthropic	✅	✅	✅	✅	✅
Cohere	✅	✅	✅	✅	✅
Google Palm	✅	✅	✅	✅	✅
Hugging Face	✅	✅	✅	✅	✅
Local Llama	✅	✅	✅	✅	✅
OpenAI	✅	✅	✅	✅	✅
…	✅	✅	✅	✅	✅

Advantages

- Common DSL/APIs
- Interoperability
- Reduced vendor lock-in
- Optionality
- Best practices
- Rich featureset

Vector search

```ruby
llm = Langchain::LLM::GooglePalm.new(api_key: "...")

client = Langchain::Vectorsearch::Milvus.new(url:, index_name:, llm: llm)

client.add_texts texts: [...], ids: [...]

my_pdf  = Langchain.root.join("/Documents/file.pdf")
my_text = Langchain.root.join("/Documents/file.txt")
my_docx = Langchain.root.join("/Documents/file.docx")

client.add_data(paths: [my_pdf, my_text, my_docx])
```

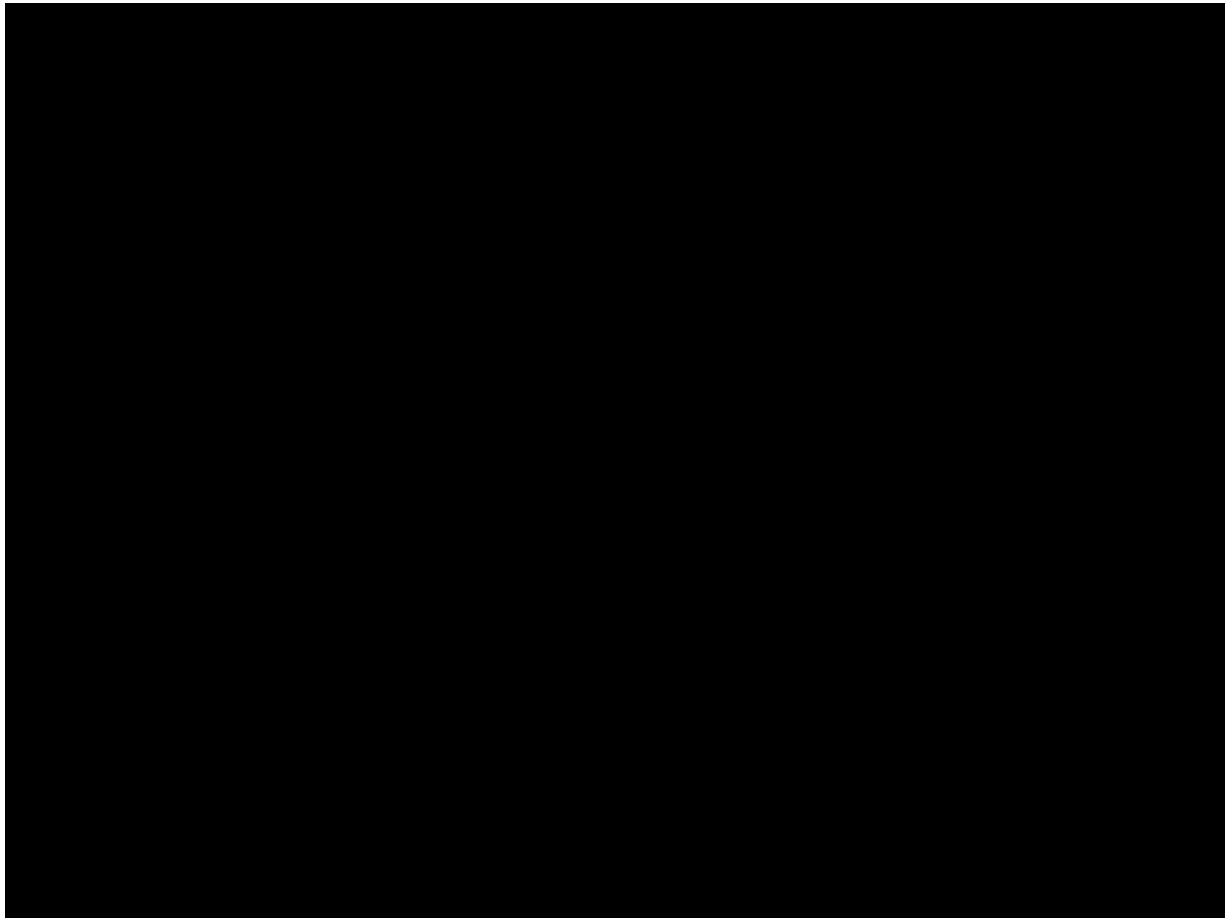
ActiveRecord integration

```ruby
class Product < ActiveRecord::Base
  vectorsearch provider: Langchain::Vectorsearch::Milvus.new(url:, index_name:, llm:)

  after_save :upsert_to_vectorsearch
end
```

Vector search

```
# Retrieve similar documents based on the query string passed in
client.similarity_search(
    query:,
    k:       # number of results to be retrieved
)

# Q&A-style querying based on the question passed in
client.ask(
    question:
)
```

Demo

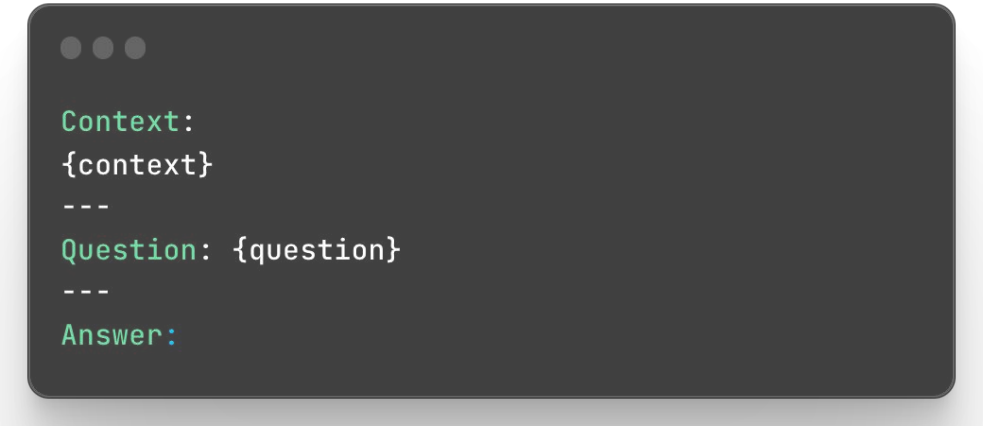


Q&A on top of imported corporate “benefits brochure” (2014)

Retrieval Augmented Generation (RAG)

Retrieving data from outside the foundation model and augmenting a prompt by adding retrieved data into the context.

1. Generate an embedding vector from the user's question
2. Similarity search against data in vector search DB (Milvus/Zilliz)
3. Insert matches into the context in the prompt
4. Prompt the LLM to generate a coherent answer



Prompt sent to the LLM

Agents (experimental)

- Autonomous general purpose LLM-powered programs
- Tons of emergent research: Chain of Thought (CoT) , Tree of Thought (ToT), Reasoning+Act (ReAct)
- Can be used to automate workflows or execute multi-step tasks
- Work better with powerful LLMs
- Driven by carefully engineered prompts
- Need lots of error handling and constraints around them (for resiliency)
- Can use “Tools” (API wrappers, similar to OpenAI Plugins)

Tools

- Calculator
- Wikipedia
- Google Search
- Weather
- Ruby Code Interpreter
- Database
- ...

Langchain::Agent::ReActAgent

Langchain::Agent::SQLQueryAgent

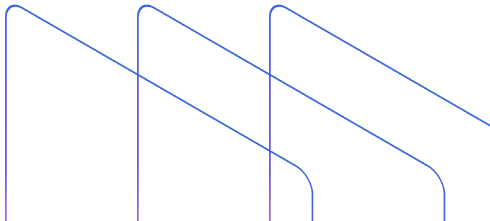
Langchain.rb Discord

Questions?

github.com/milvus-io/milvus

FAQ Topics

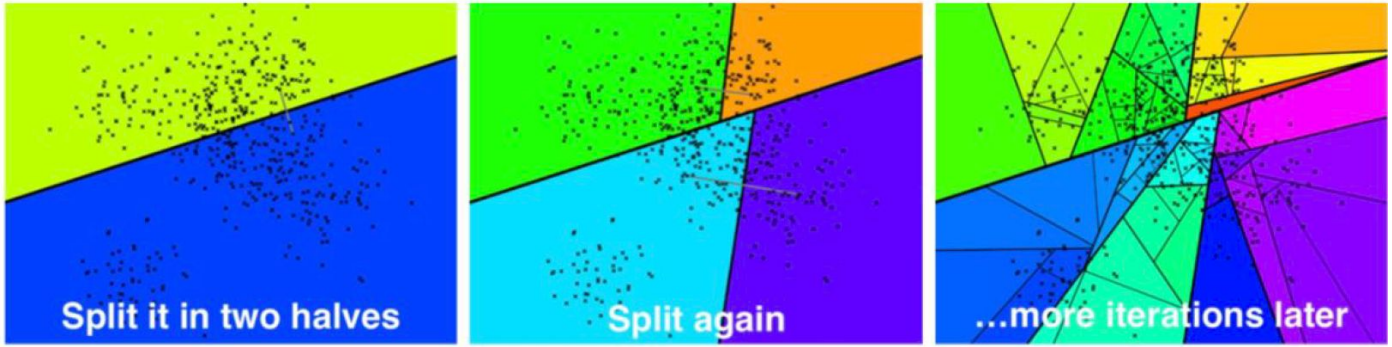
- When *NOT* to use
- CSV Files
- Role Based Access Control
- Scaling Up and Down
- Hybrid Search



05

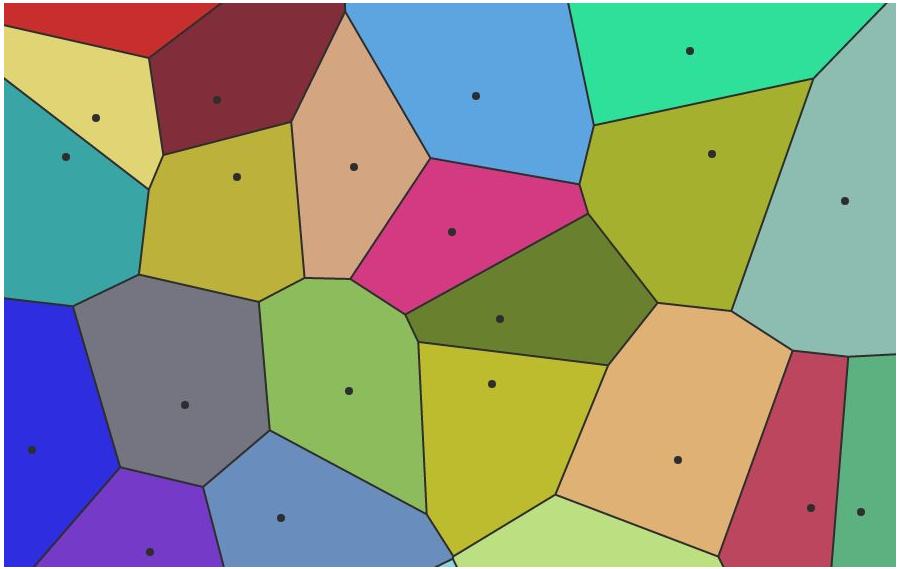
Vector Database FAQ

Approximate Nearest Neighbors Oh Yeah



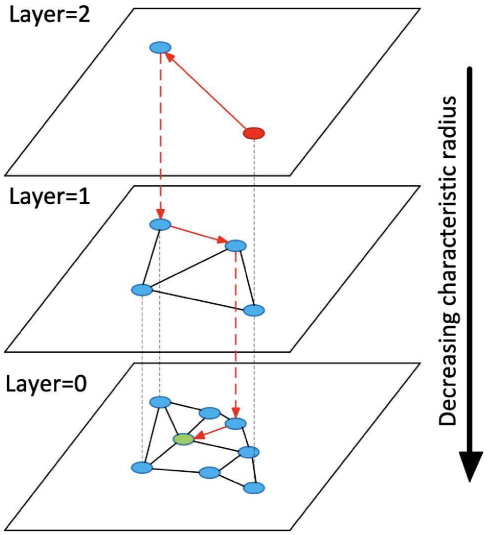
Source:
https://sds-aau.github.io/M3Port19/portfolio/ann/

Inverted File Index

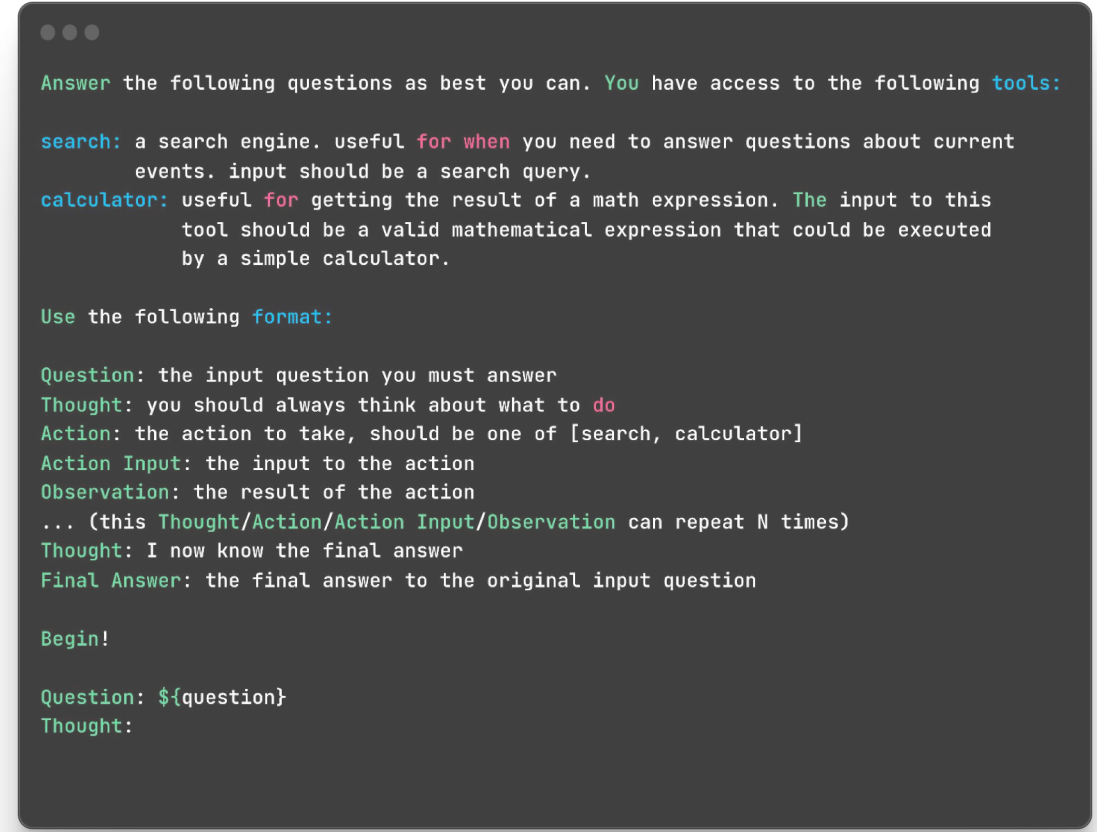


Source:
https://towardsdatascience.com/similarity-search-with-ivfpq-9c6348fd4db3

Hierarchical Navigable Small Worlds (HNSW)



Source:
https://arxiv.org/ftp/arxiv/papers/1603/1603.09320.pdf



```
Answer the following questions as best you can. You have access to the following tools:

search: a search engine. useful for when you need to answer questions about current
        events. input should be a search query.
calculator: useful for getting the result of a math expression. The input to this
            tool should be a valid mathematical expression that could be executed
            by a simple calculator.

Use the following format:

Question: the input question you must answer
Thought: you should always think about what to do
Action: the action to take, should be one of [search, calculator]
Action Input: the input to the action
Observation: the result of the action
... (this Thought/Action/Action Input/Observation can repeat N times)
Thought: I now know the final answer
Final Answer: the final answer to the original input question

Begin!

Question: ${question}
Thought:
```

Prompt used in Langchain::Agent::ReActAgent