Weaviate

Bring AI-based search to your web app

JP Hwang
Educator

(**@_jphwang** on Threads / Twitter)

TECH WORLD IN 2023
DOES A THING
IS THIS AI?
imgflip.com

Agenda

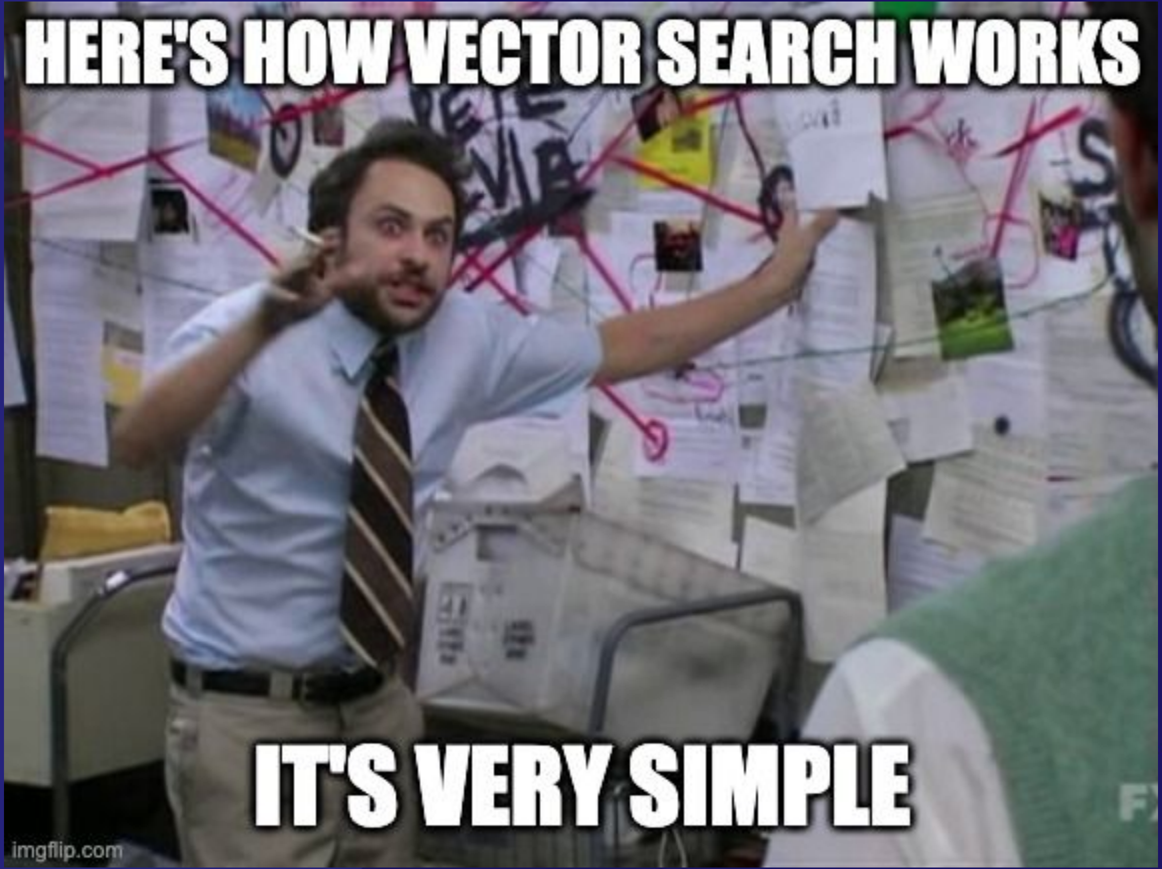
- Demo: What you can do with “AI” (vector) search
- Behind the magic - how does it work?
- Demo: AI search + large language model (LLM)

Teach specific syntax
De-mystify "AI" tech
imgflip.com

Agenda

- Demo: What you can do with “AI” (vector) search
- Behind the magic - how does it work?
- Demo: AI search + large language model (LLM)

Goal: To convince you that adding “AI” to your tech stack / web app is very achievable.


HERE'S HOW VECTOR SEARCH WORKS
IT'S VERY SIMPLE
imgflip.com

Demo: Semantic search

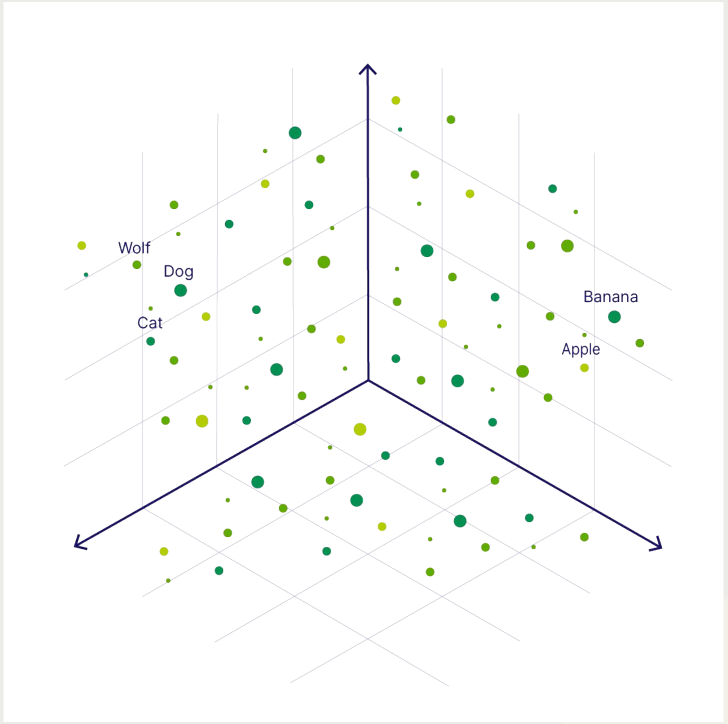
“To get good results, you shouldn’t need to know any magic words. With semantic search, you don’t.”

- David Haney, David Gibson

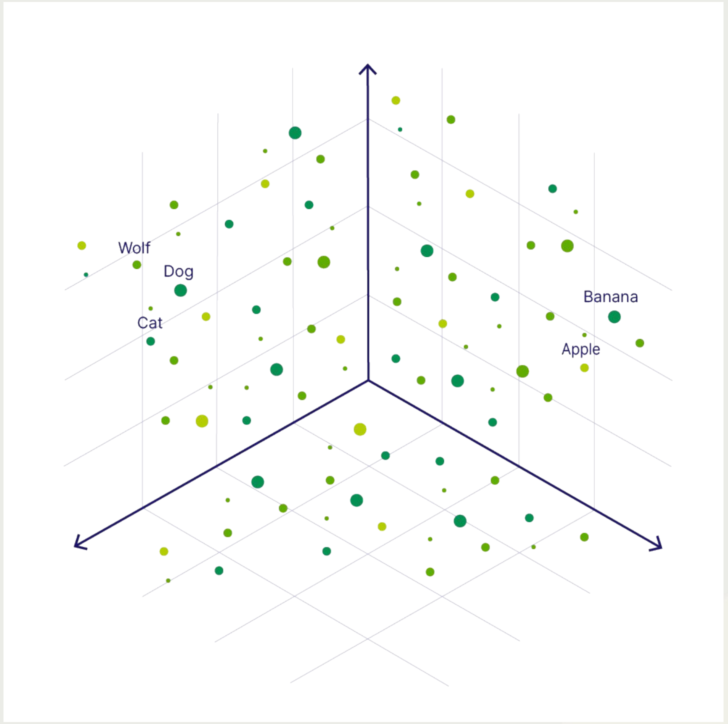
Stackoverflow Blog

How does semantic search work?

Meaning is “vectorized”

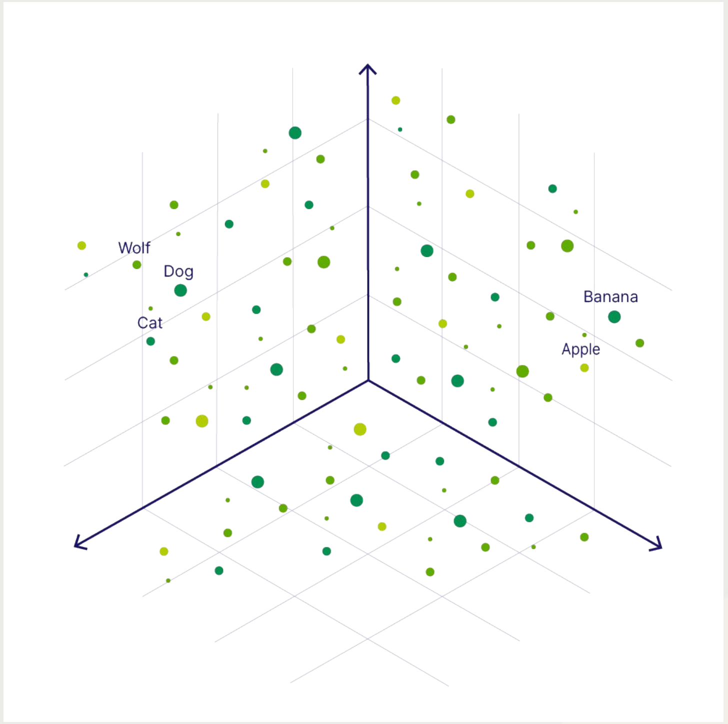


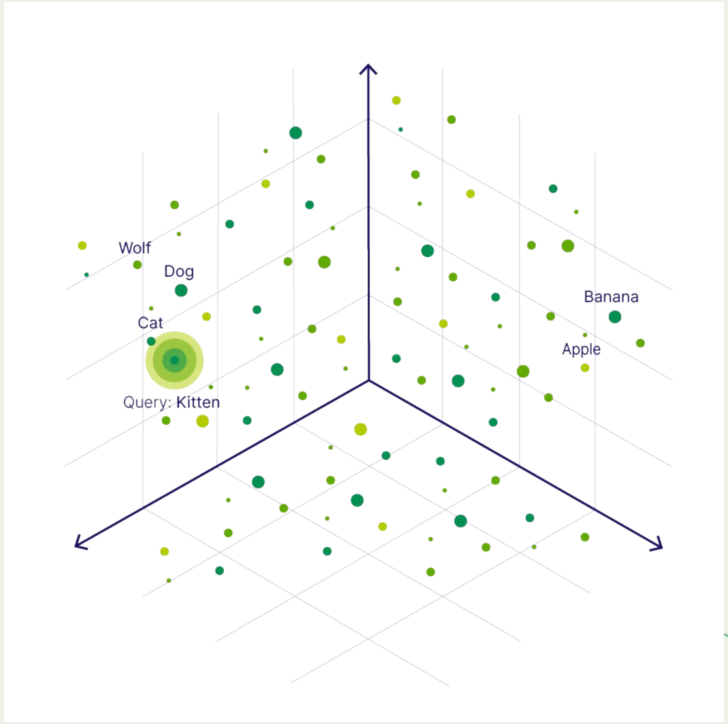
Meaning is “vectorized”



And now we can perform similarity searches

Meaning is “vectorized”





How do vectors work?

Let’s step back a bit…

Are there other areas where we quantify similarity?

Let’s step back a bit…

Are there other areas where we quantify similarity?

Consider *Colors*.

Let’s step back a bit…

Are there other areas where we quantify similarity?

Consider *Colors*.

Have you used the RGB system?
(255, 0, 0) = red
(80, 200, 120) = emerald.

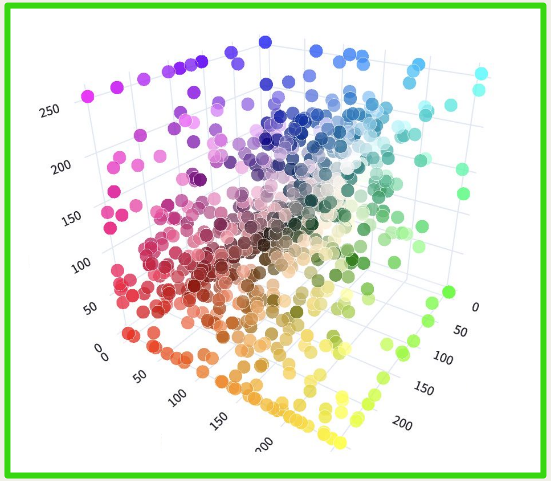
Let’s step back a bit…

Are there other areas where we quantify similarity?

Consider *Colors*.

Have you used the RGB system?
(255, 0, 0) = red
(80, 200, 120) = emerald.

This puts similar colors closer in space.



Now extend this concept…

To hundreds, or even thousands of numbers.

We can represent complex meaning this way.

Example

- "Three people rescued off Australian coast ..."

Vector

```
[-0.01670855, -0.02290458,
0.01024679, ..., -0.01840662,
-0.01677336,  0.00040852]
```

Examples

- "Three people rescued off Australian coast ..."
- "Tourists taking selfies and feeding dingoes ..."
- "Sam Kerr: Chelsea striker and Matildas captain ..."
- "'She's brilliant': Mary Earps inspires girls ..."

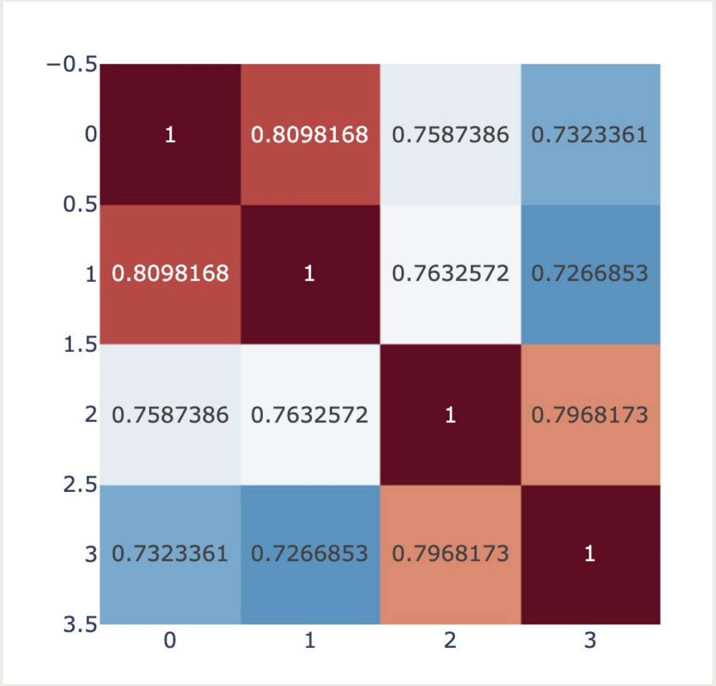
Vectors

```
[-0.01670855, -0.02290458,
0.01024679, ..., -0.01840662,
-0.01677336,  0.00040852]

[-0.01062017,  0.01388064,
0.02811302, ..., -0.01565292,
0.00282415, -0.01064047]

[-0.00067538, -0.00483041,
0.02590884, ..., -0.01845455,
-0.01025612, -0.00987435]

[-0.03254206,  0.00462641,
0.00465651, ...,  0.01225011,
-0.00032469, -0.01669922]
```

Examples

- "Three people rescued off Australian coast ..."
- "Tourists taking selfies and feeding dingoes ..."
- "Sam Kerr: Chelsea striker and Matildas captain ..."
- "'She's brilliant': Mary Earps inspires girls ..."

Similarity matrix



	0	1	2	3
0	1	0.8098168	0.7587386	0.7323361
1	0.8098168	1	0.7632572	0.7266853
2	0.7587386	0.7632572	1	0.7968173
3	0.7323361	0.7266853	0.7968173	1

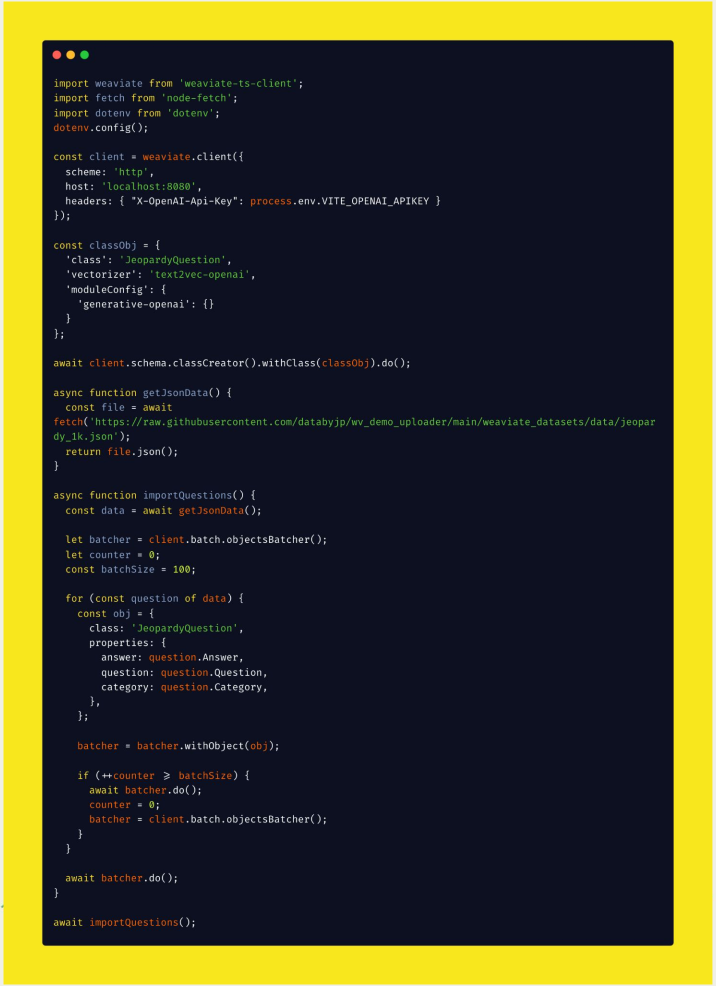
This is how “vector search” works

- Objects ➡️ vectors ➡️ similarity search
- Enabled by modern deep learning models
- Vector DBs index data by vector

How hard it to use?

- Node.js script:
  - ~50 lines of code
  - Objects ➡️ vectors

    @ import time

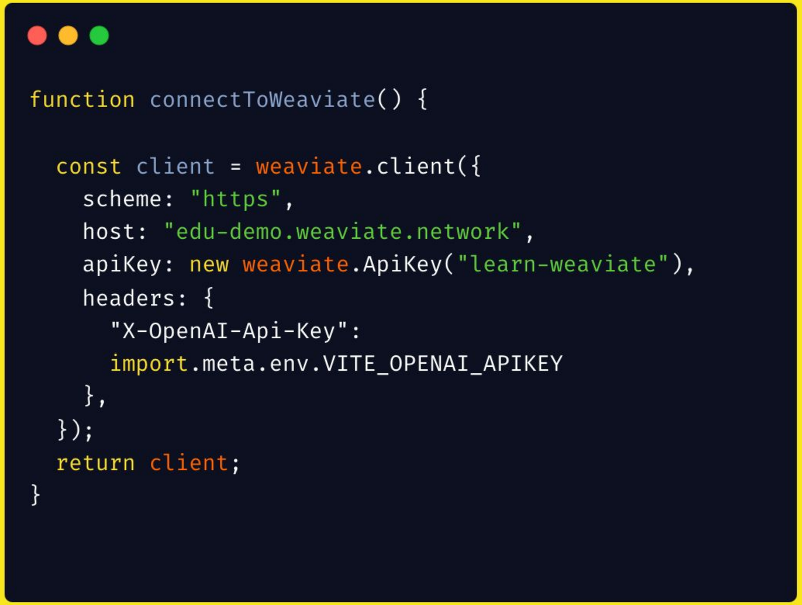


```
import weaviate from 'weaviate-ts-client';
import fetch from 'node-fetch';
import dotenv from 'dotenv';
dotenv.config();

const client = weaviate.client({
  scheme: 'http',
  host: 'localhost:8080',
  headers: { "X-OpenAI-Api-Key": process.env.VITE_OPENAI_APIKEY }
});

const classObj = {
  'class': 'JeopardyQuestion',
  'vectorizer': 'text2vec-openai',
  'moduleConfig': {
    'generative-openai': {}
  }
};

await client.schema.classCreator().withClass(classObj).do();

async function getJsonData() {
  const file = await
fetch('https://raw.githubusercontent.com/databyjp/wv_demo_uploader/main/weaviate_datasets/data/jeopar
dy_1k.json');
  return file.json();
}

async function importQuestions() {
  const data = await getJsonData();

  let batcher = client.batch.objectsBatcher();
  let counter = 0;
  const batchSize = 100;

  for (const question of data) {
    const obj = {
      class: 'JeopardyQuestion',
      properties: {
        answer: question.Answer,
        question: question.Question,
        category: question.Category,
      },
    };

    batcher = batcher.withObject(obj);

    if (++counter ≥ batchSize) {
      await batcher.do();
      counter = 0;
      batcher = client.batch.objectsBatcher();
    }
  }

  await batcher.do();
}

await importQuestions();
```

How hard it to use?

- React front end:
  - Connect

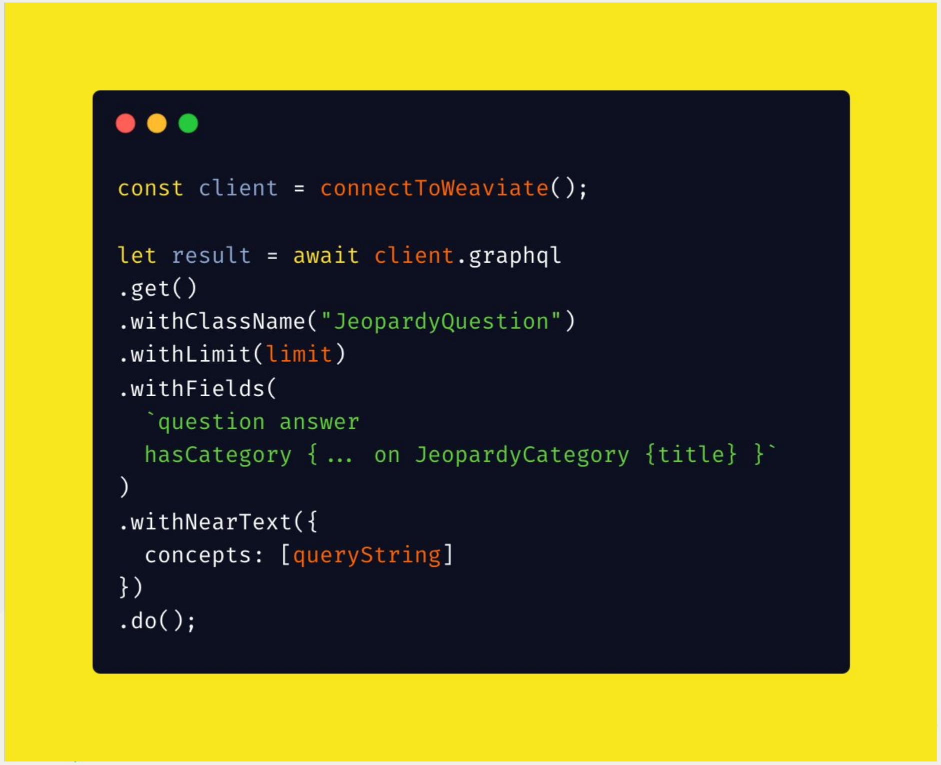
    (read-only key)



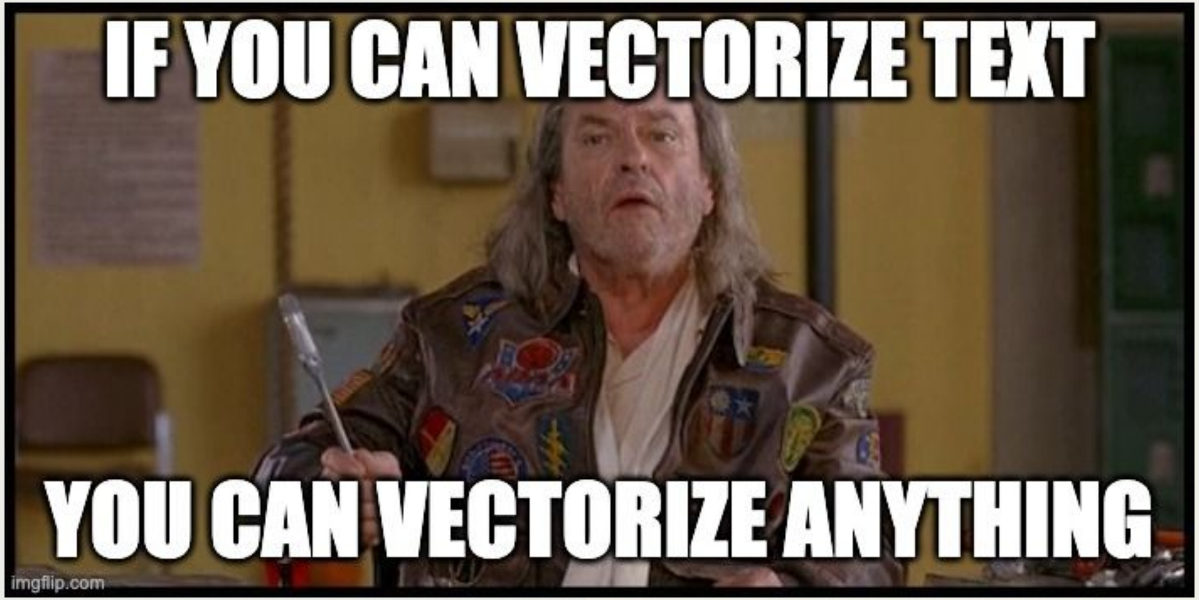
```
function connectToWeaviate() {

  const client = weaviate.client({
    scheme: "https",
    host: "edu-demo.weaviate.network",
    apiKey: new weaviate.ApiKey("learn-weaviate"),
    headers: {
      "X-OpenAI-Api-Key":
      import.meta.env.VITE_OPENAI_APIKEY
    },
  });
  return client;
}
```

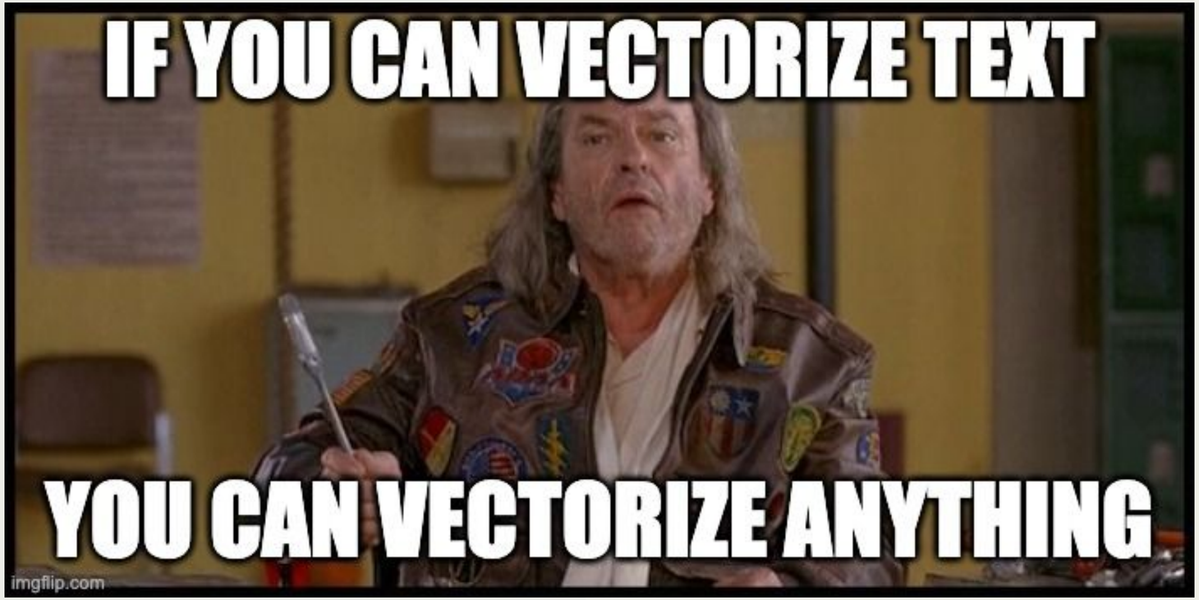
How hard it to use?

- React front end:
    - Query:



```
const client = connectToWeaviate();

let result = await client.graphql
.get()
.withClassName("JeopardyQuestion")
.withLimit(limit)
.withFields(
  `question answer
  hasCategory { ... on JeopardyCategory {title} }`
)
.withNearText({
  concepts: [queryString]
})
.do();
```

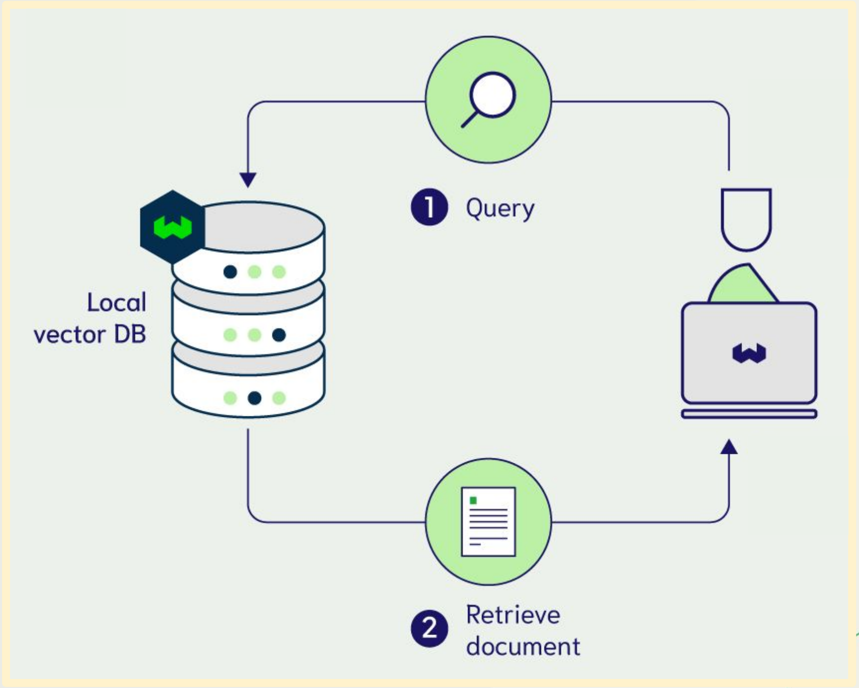

IF YOU CAN VECTORIZE TEXT
YOU CAN VECTORIZE ANYTHING
imgflip.com



Some models can even vectorize

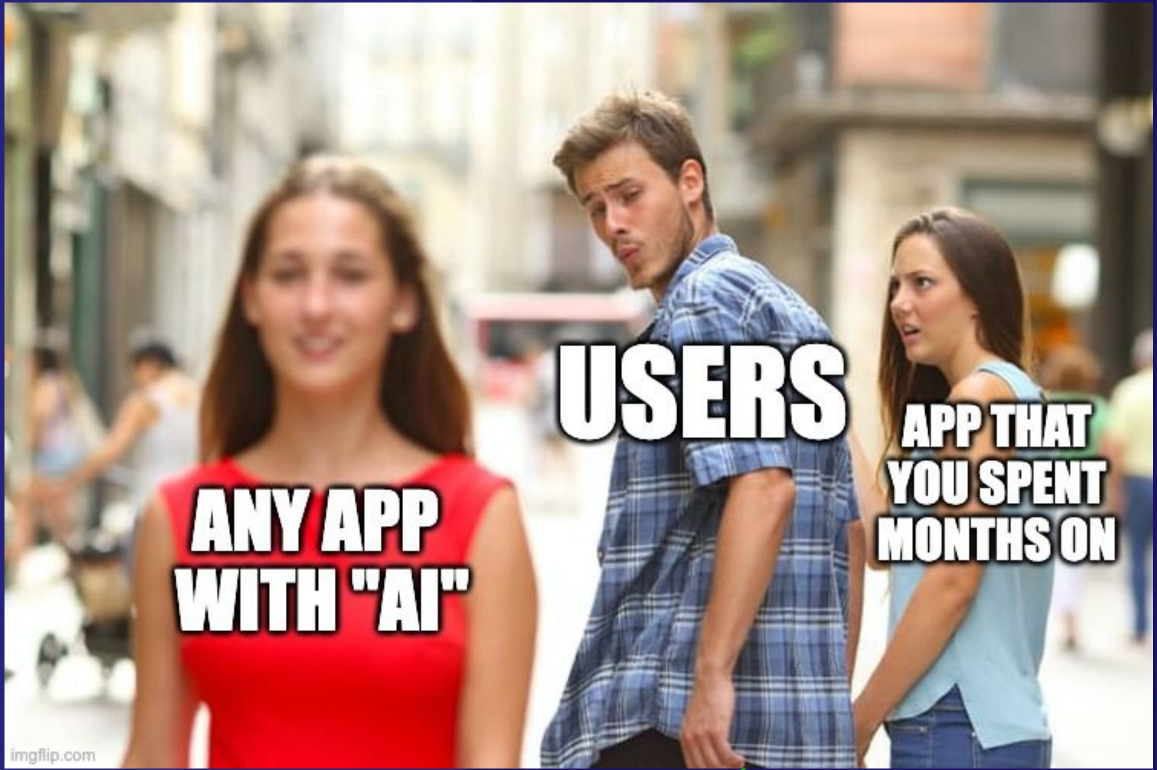
images, video, text, IMU, thermal data…

Vector search



Straw poll:
Have you used ChatGPT?

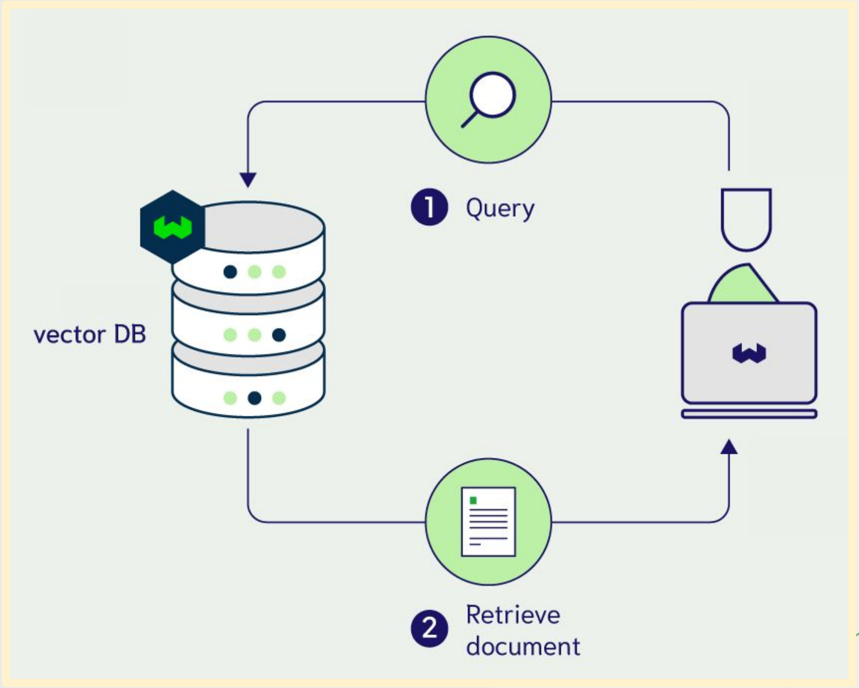
yes / yes

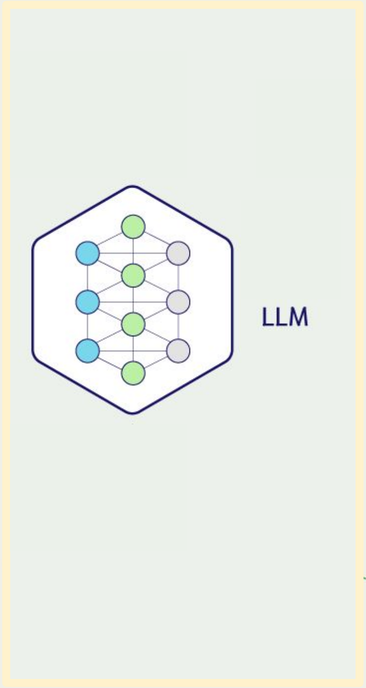

ANY APP WITH "AI"
USERS
APP THAT YOU SPENT MONTHS ON
imgflip.com

LLM + your data = user ❤️

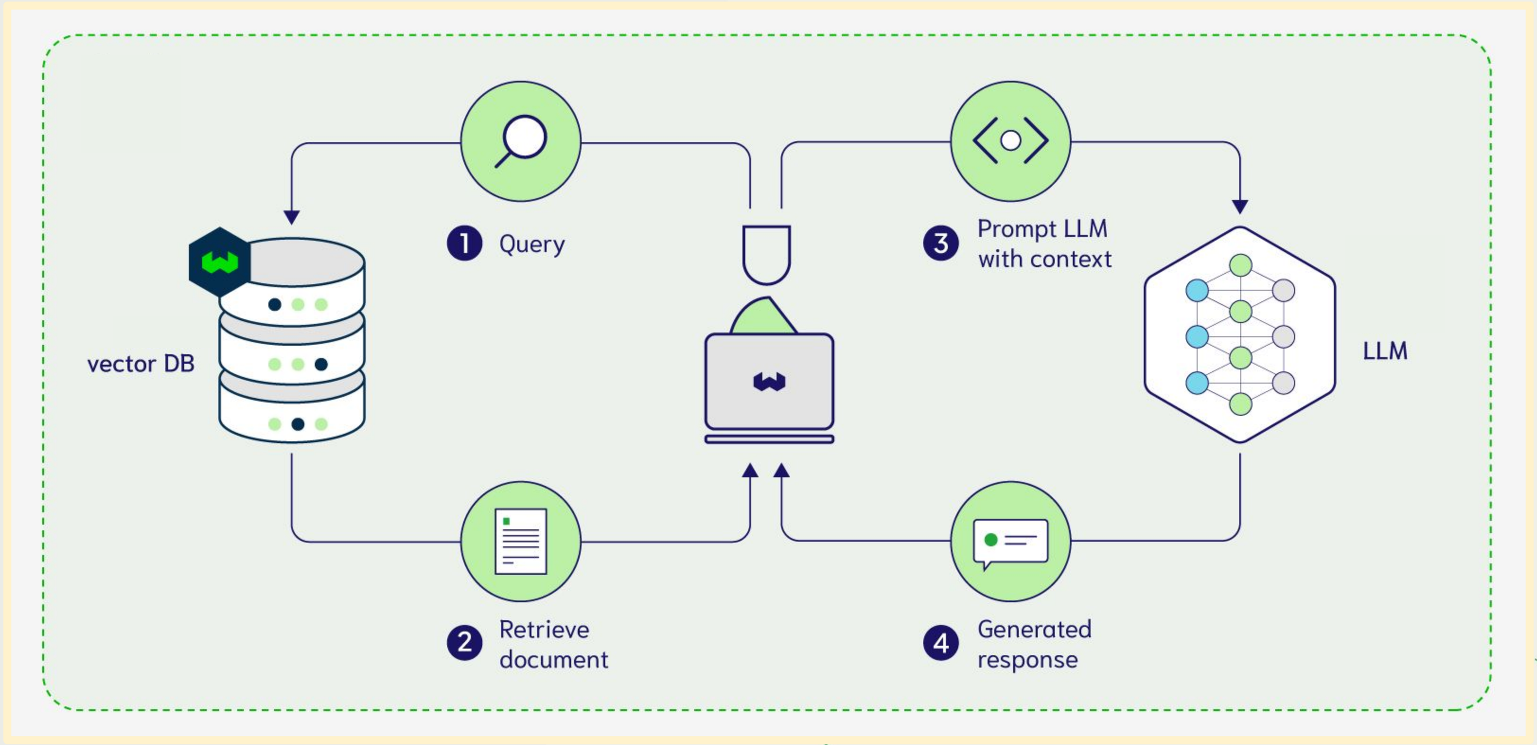
Demo: Retrieval augmented generation

Vector search + LLM





Retrieval augmented generation



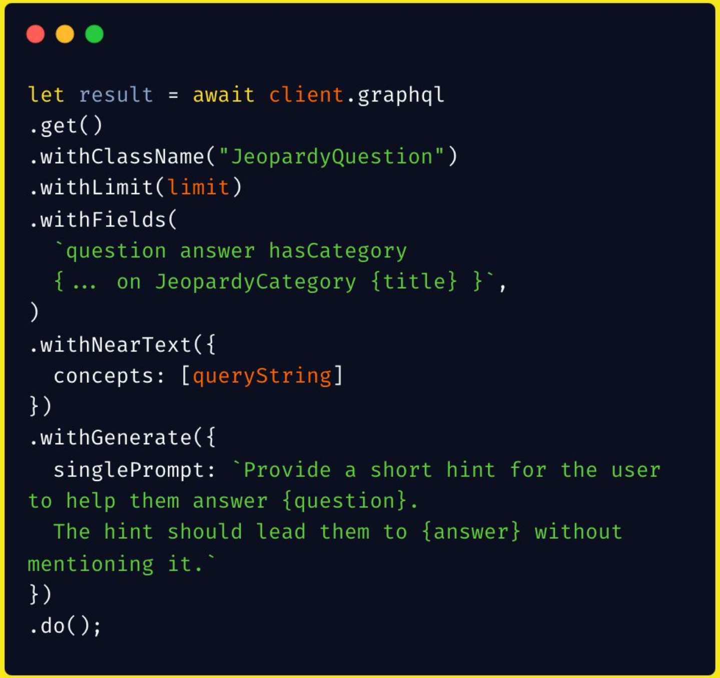
Retrieval augmented generation

- Retrieves data
- Sends the data+prompt to an LLM
- Serves data + LLM response

(Some of the served outputs are not in the DB!)

How hard it to use?

- React front end:
  - Query:



```
let result = await client.graphql
.get()
.withClassName("JeopardyQuestion")
.withLimit(limit)
.withFields(
  `question answer hasCategory
  { ... on JeopardyCategory {title} }`,
)
.withNearText({
  concepts: [queryString]
})
.withGenerate({
  singlePrompt: `Provide a short hint for the user
to help them answer {question}.
  The hint should lead them to {answer} without
mentioning it.`
})
.do();
```

Bonus Demo

Recap

Vector databases provide **AI-first tooling**

Recap

Vector databases provide **AI-first tooling** to make your life easier as a builder.

Recap

Vector databases provide **AI-tooling**

- Vector searches.
  - Semantic to multi-modal
- LLM integration.
- Scale easily to production.

Codebase +

Further resources:

Thank you

Connect with us!

weaviate.io

weaviate/weaviate

weaviate_io