

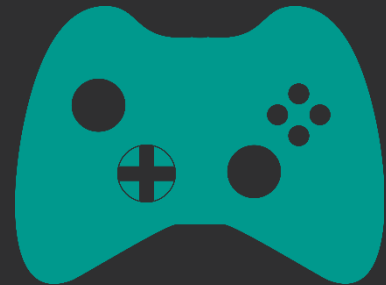


A **fun** and **absurd** introduction to **Vector Databases**

ALEXANDER CHATZIZACHARIAS
@glycin.bsky.social



PRESENTER
ALEXANDER CHATZIZACHARIAS



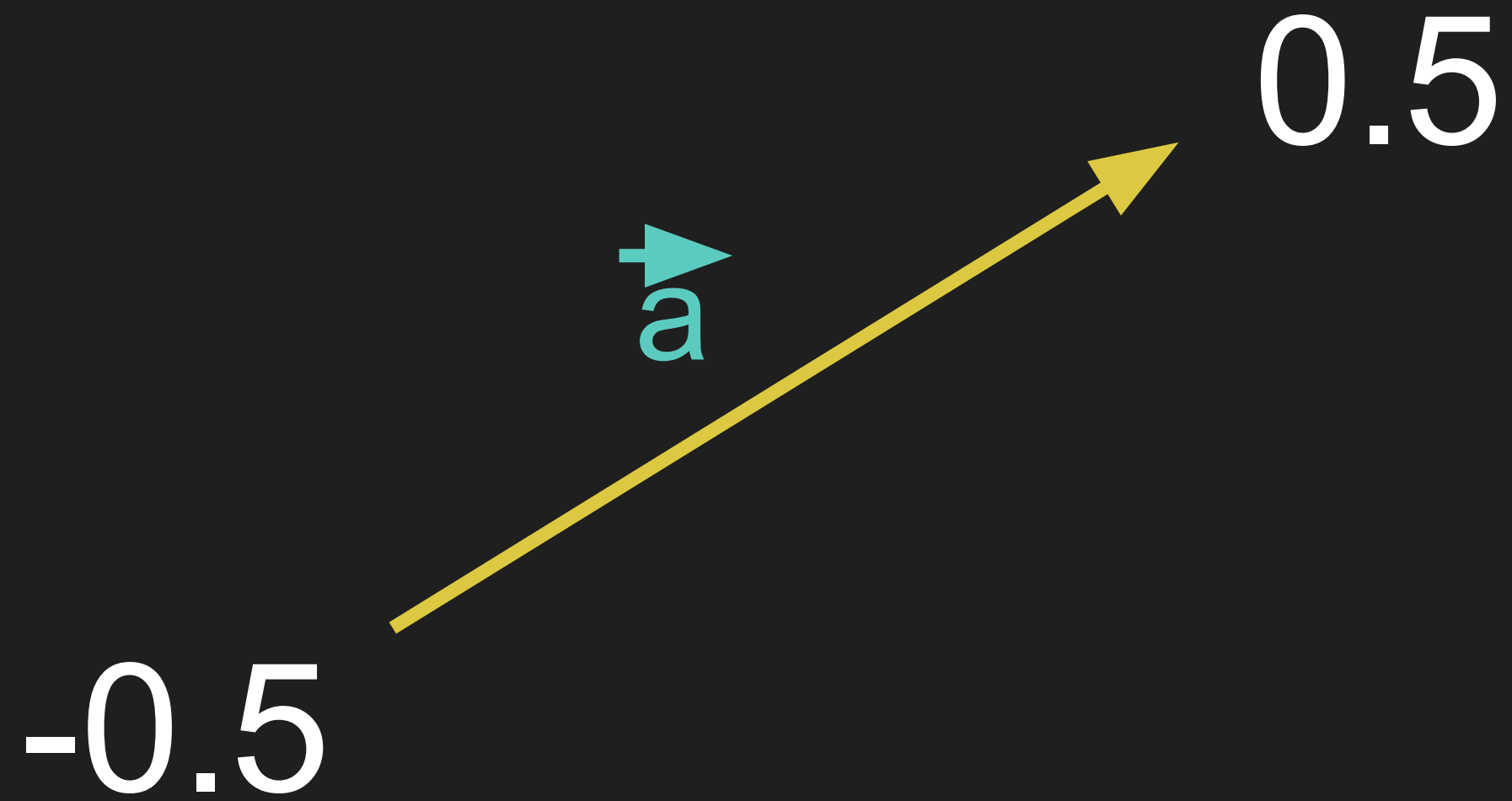


GOAL OF THIS TALK



VECTORS

What are **vectors**?



What are **vectors**?



[0.5, -0.5, 0]

dimensions = 3

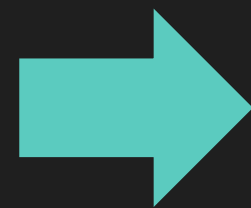
[0.5, -0.5, 1.0, 0.3, -0.3, 0.9, -0.9, 0.01]

dimensions = 8

How do we use vectors?



How many **software developers** does it take to screw in a **lightbulb**?



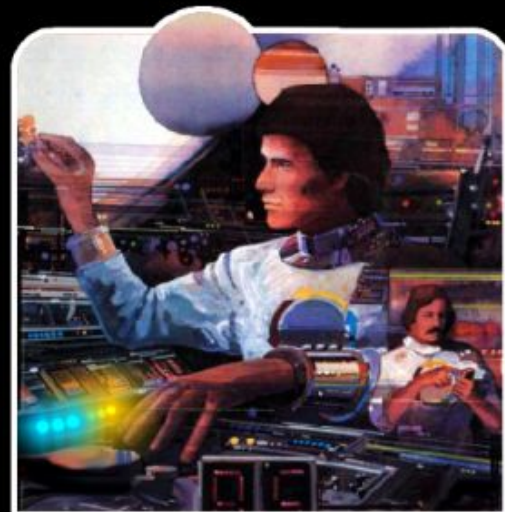
None, it's a hardware problem

`[-0.006748975,-0.023399254,0.004803278,
-0.03715345,0.018772759,-0.01941031,-0.
003221868,-0.013975525,0.017044844,0.0
17906666,0.043792102,-0.0033201252,-0.
014332535,-0.0029526732,0.032242633,0.
025404716,...]`

How do we use vectors?



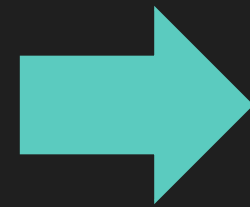
THE TWO STATES OF EVERY PROGRAMMER



I AM A GOD.



**I HAVE NO IDEA
WHAT I'M DOING.**



[0.026854644,0.046787556,0.0002151226,
0.037302915,0.009727323,0.0042568287,
0.04576964,0.0013366787,-0.013471359,0
.017539714,-0.010231924,0.050567146,-0.
050488357,-0.039395656,-0.016778067,0.
025073335,-0.014173437,-0.044925395,0.
019400975,-0.014789729,-0.005302614,-0.
0033920964,0.040046062,0.0005339275,-
0.018291568,-0.058520786,-0.034676168,
0.015270221,-0.0033610577,0.027547115,
-0.025901454,0.00091994856,...]

MACHINE LEARNING

In particular, **vectorization** techniques (or **embedding** models)

Some **Vectorization** techniques



word2vec



GloVe

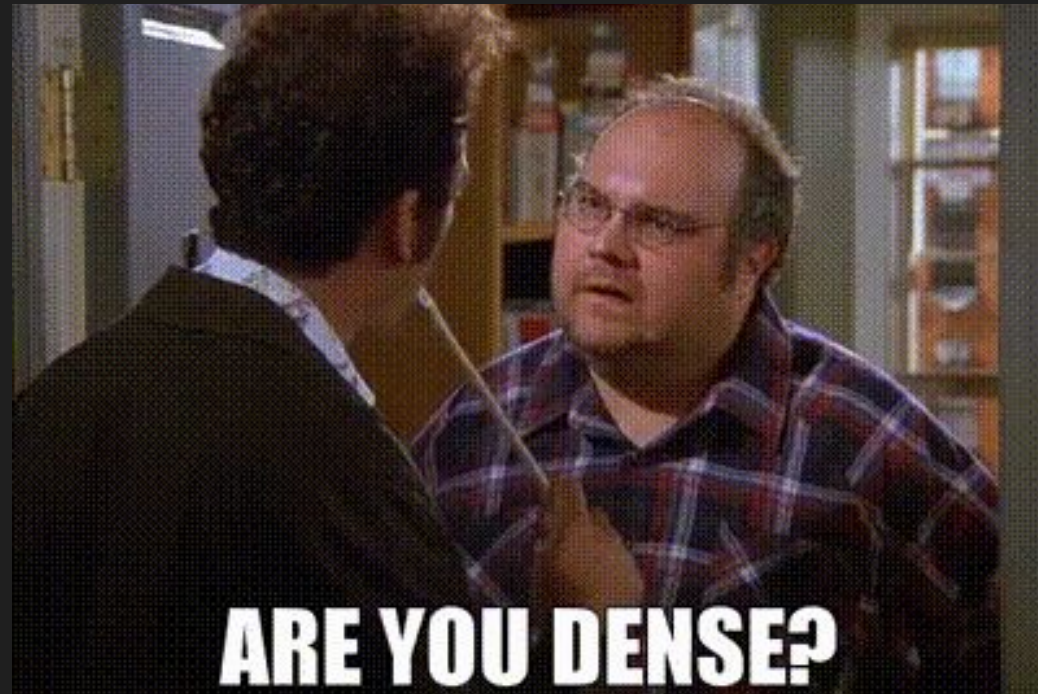


BERT



CLIP

Types of vector representations



Dense



Sparse

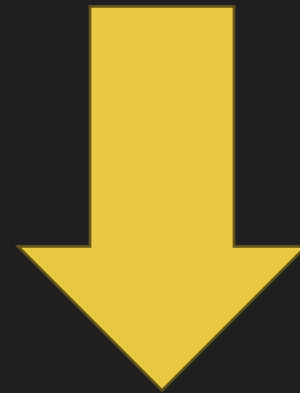


Multi-vector

What are **sparse vectors**?



High dimensional vectors with a lot of 0 values. Typically stored as **key-value pairs**

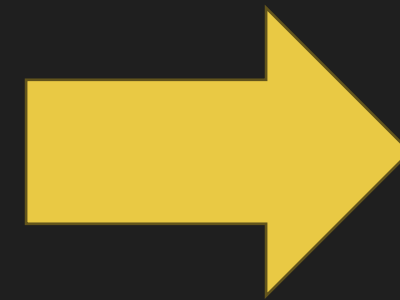


[{5:8}, {10:2}, {13:4}]

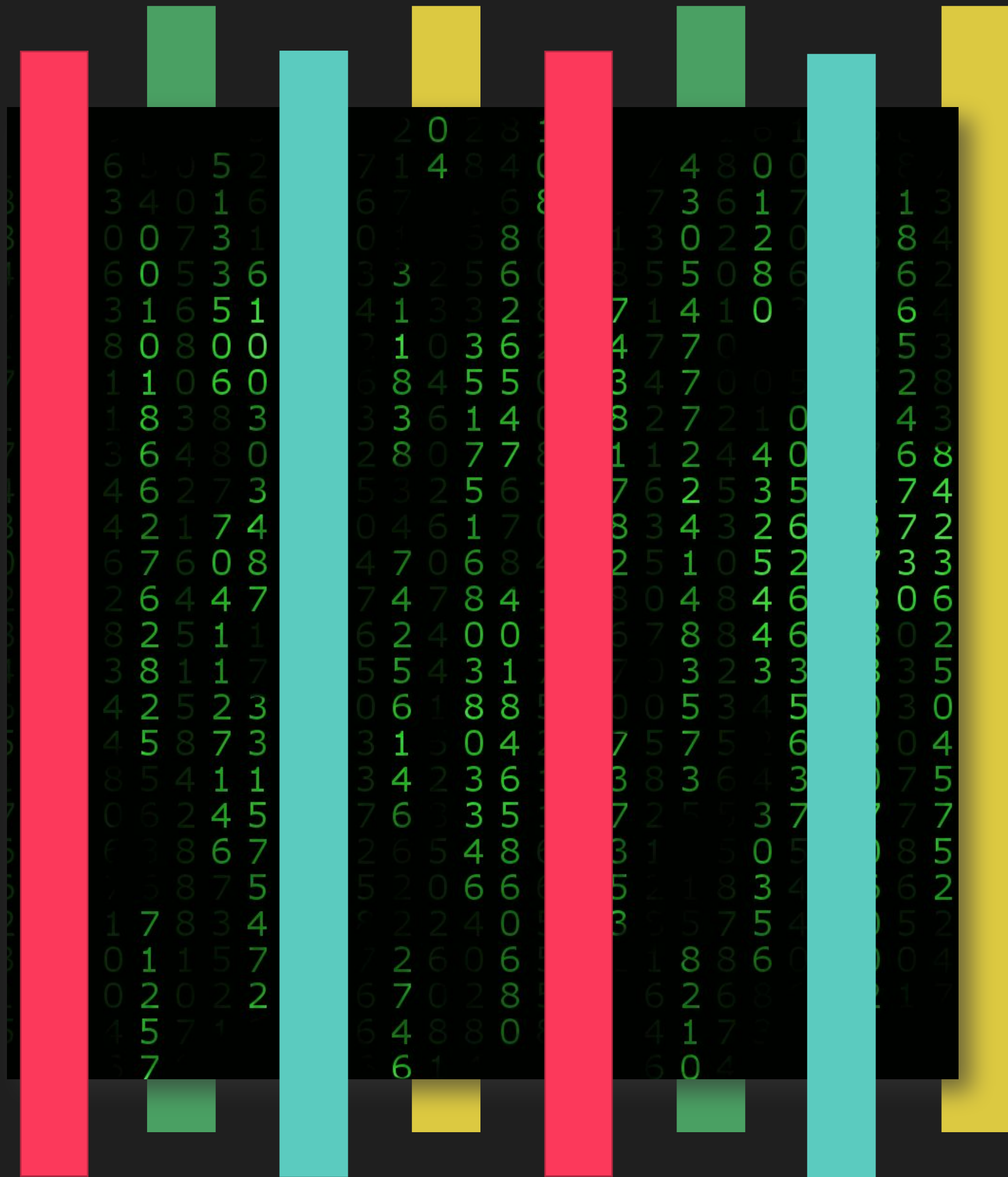
What are multi-vectors?



Generated using
embedding models.
Hold high semantic
meaning, but require
more computation



[
[0.5, -0.5, 1.0],
[0.3, -0.1, 0.9],
[0.2, -0.3, 0.1],
[0.1, -0.5, 0.7],
]



VECTOR DATABASES

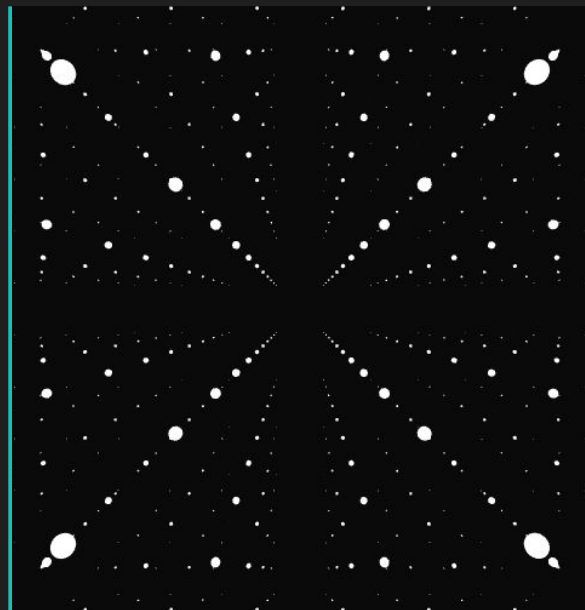
What is a **Vector** Database?



Purpose built database



Retrieval using **Nearest Neighbor** search (NNS)



Store & index **high dimensional** vectors



Store **metadata** associated with vector objects

A **database** facilitating
effortless and swift
semantic search on **your**
data.

Artificial Intelligence



Indexing in Vector Databases

Exact
Nearest Neighbor

linear search

k-nearest neighbors

space **partitioning**

...

Approximate
Nearest Neighbor

Inverted file with flat compression (**IVFFlat**)

Locality-sensitive hashing (**LSH**)

Approximate Nearest Neighbors Oh yeah (**ANNOY**)

Hierarchical Navigable Small World (**HNSW**)

...

Distance metrics



Euclidean distance / similarity

Cosine distance / similarity

Hamming distance

Manhattan distance

Dot product

Vector Space

Distance calculations and Indexing always happen
by schema by table by object, and more...
within the same "Vector Space"

Vector Dimensions = 512



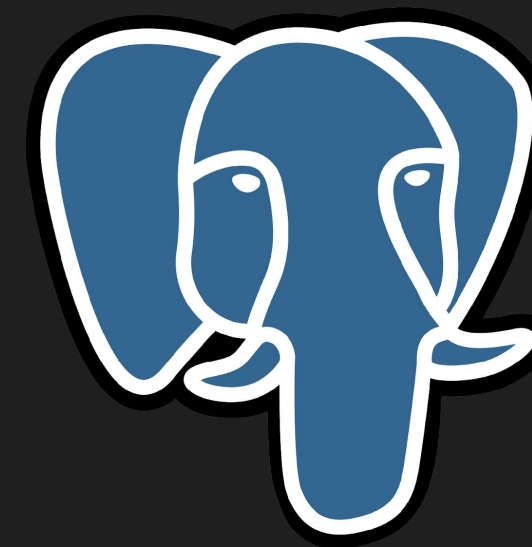
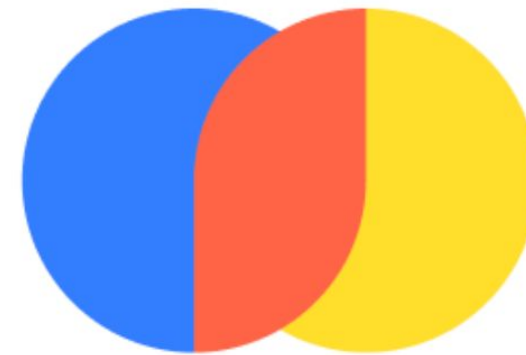
Vector Dimensions = 384

Vector Dimensions = 384



Vector Dimensions = 384

Vector Databases



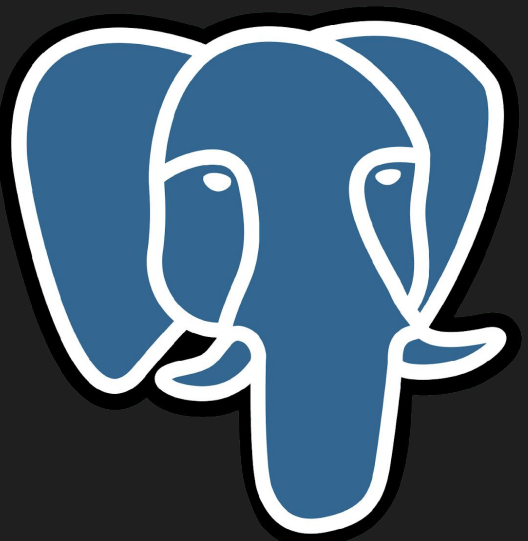
Vector Databases



Closed

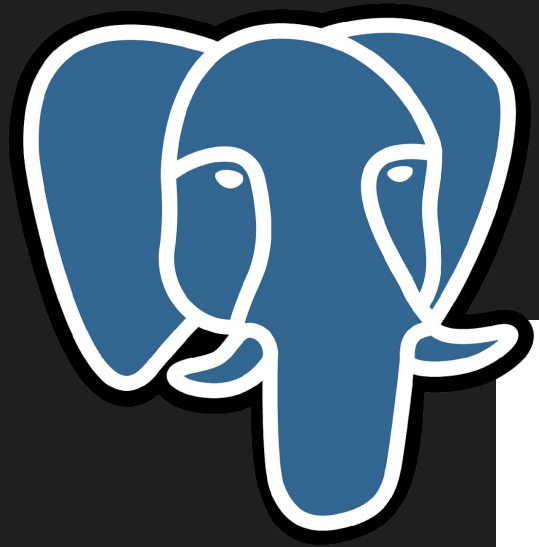


Open



Vector Databases

No built-in embeddings



elastic

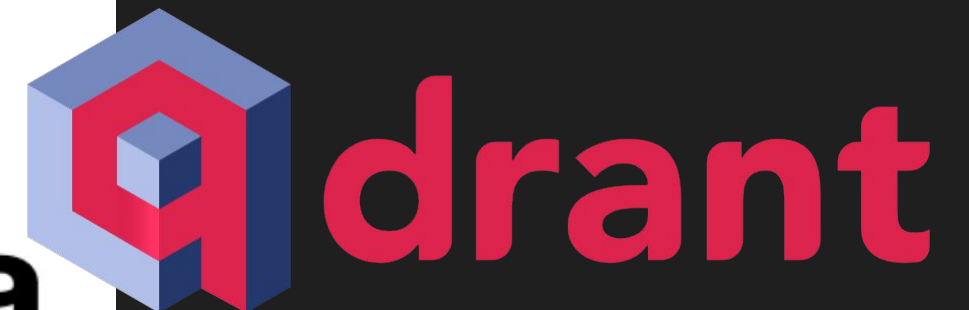
Built-in embeddings



vespaaviate



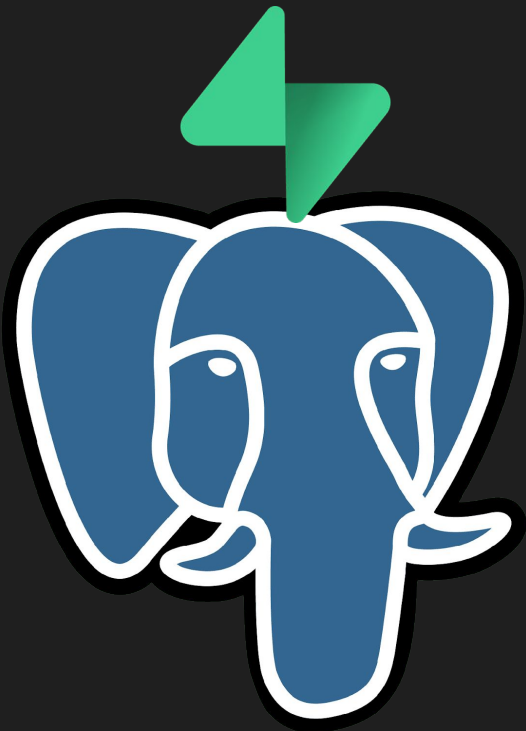
Chroma



Vector Databases



Unmanaged



Managed



milvus



vespa



Pinecone



elastic



Chroma



Weaviate



drant



redis

Vector Databases



redis

Python

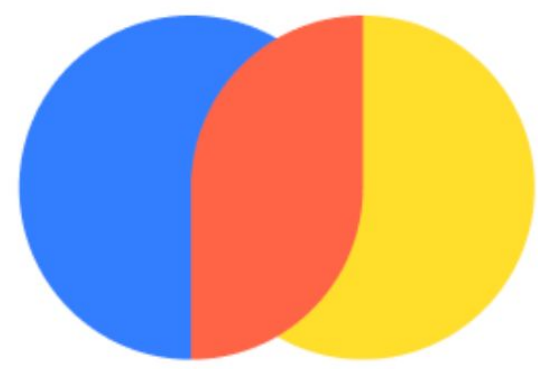


textai

REST



Pinecone



Python/ JS

Chroma

REST

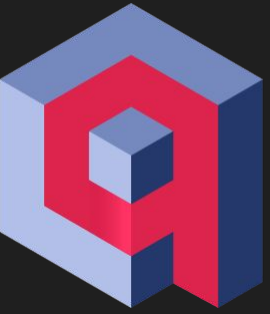


Weaviate

REST API



milvus



drant

REST API



elastic



SQL



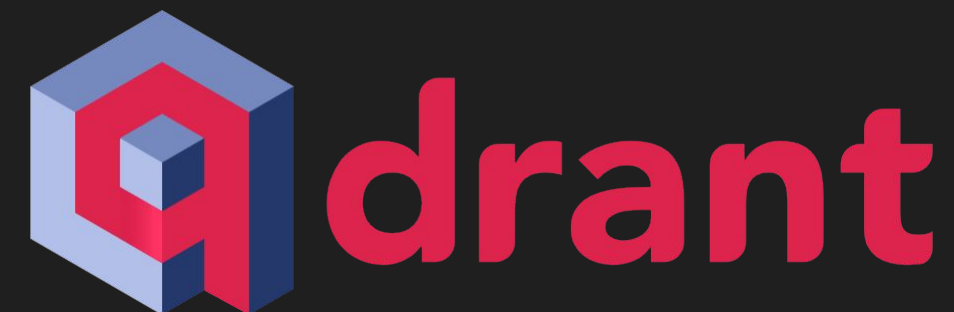
vespa

YQL

Vector Databases



Or by their **official** &
unofficial SDK's



Vector Databases



Google Cloud



Implementing our own search



Our stack



Persistence



Backend



Frontend

Why Weaviate?



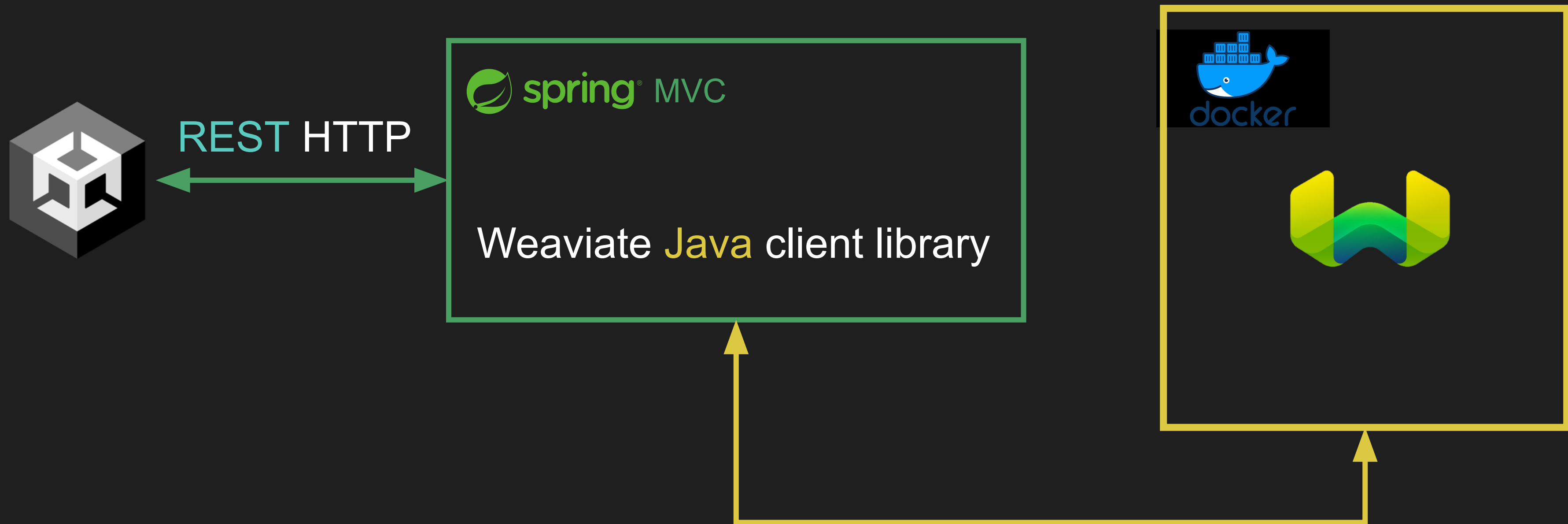
Open source, paid cloud services

Modular, can run vectorization modules alongside the DB using one simple Dockerfile

Decent documentation

Decent Java client

Architecture



Object insertion => REST HTTP

Object retrieval => GraphQL HTTP

The focus for today
●●●●

Cosine Distance as distance metric

HNSW as indexing algorithm

BERT model for text vectorization

RESNET50 model for image vectorization

CLIP model for same space vectorization

Cosine distance

$$\text{Cosine Distance} = 1 - \text{Cosine Similarity}$$

$$\text{Cosine Similarity} = \frac{\text{A} \cdot \text{B}}{(\text{length}(\text{A}) * (\text{length}(\text{B})))}$$

Dot product

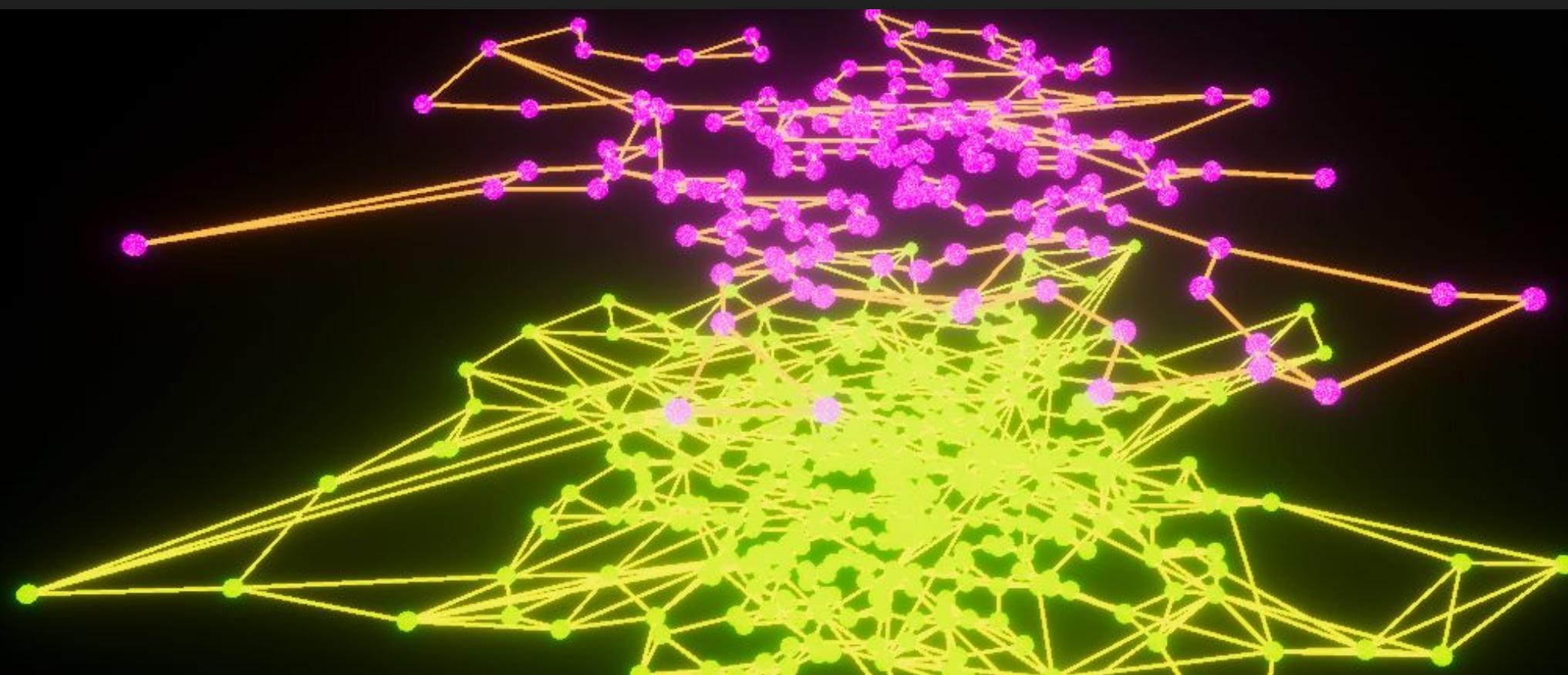
Cross product

0.0 .. 2.0

Identical

Opposite

Hierarchical Navigable Small World (HNSW)



Guns, lots of guns...





guns, lots of guns

weapons.csv



weapon name

weapon type

weapon range



Name = Excalibur

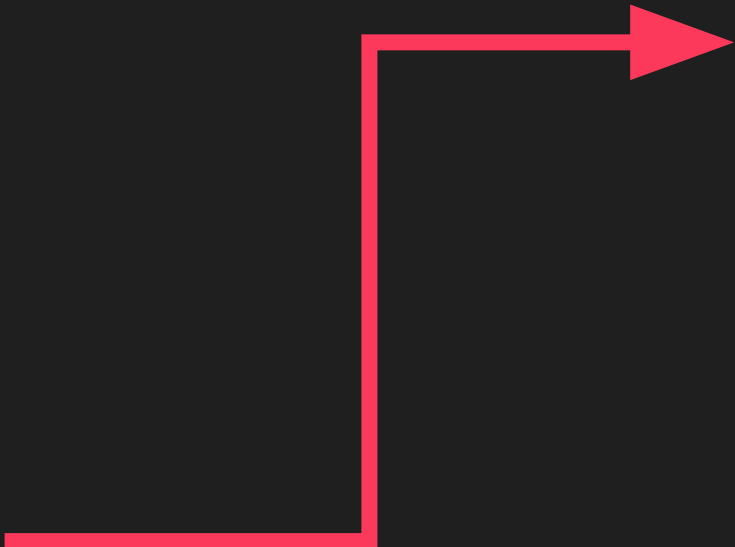
Type = Sword

Range = Short

Name = Excalibur

Type = Sword

Range = Short



weapon excalibur short sword



BERT

[-0.006748975,-0.023399254,0.004803278,-0.03715345,
0.018772759,-0.01941031,-0.003221868,-0.013975525,0
.017044844,0.017906666,0.043792102,-0.0033201252,-
0.014332535,-0.0029526732,0.032242633,0.025404716,
...]

384 dimensions

A wizard... ambushed!





BERT



Pokémon photography!





Pokémon SnapTM

PRESS START

©1995,1996,1998 Nintendo/Creatures/GAME FREAK
©1999 Nintendo/NAL Laboratory, Inc.



RESNET50

You found **Vectorichu!**

A wizard ambushed... AGAIN?!





BERT



CLIP



Do you folks like coffee?





**THANKS FOR
WATCHING**

DOWNLOAD & USE **THE GOTO GUIDE APP**

DOWNLOAD THE APP
CREATE YOUR SCHEDULE
ASK QUESTIONS
RATE SESSIONS





Q &

A