Data Days

Under the hood of vector databases

By Aurélien Massiot

About me

that's a Jerusalem artichoke

Aurélien

Aurélien, a rare animal, has been an AI expert since his arrival at OCTO Technology more than 7 years ago.

His leitmotifs?

Simplicity, pragmatism.

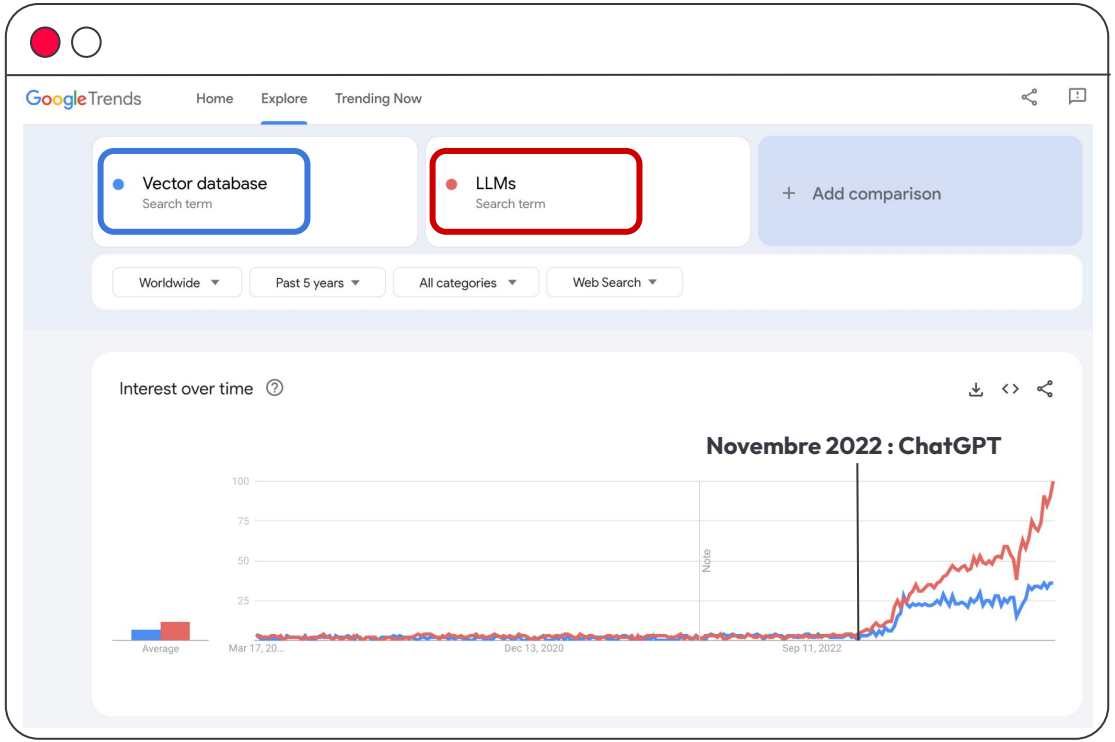
His passions

- Traveling to distant lands
- Photography
- Jerusalem artichokes

I had my first photo exhibition!

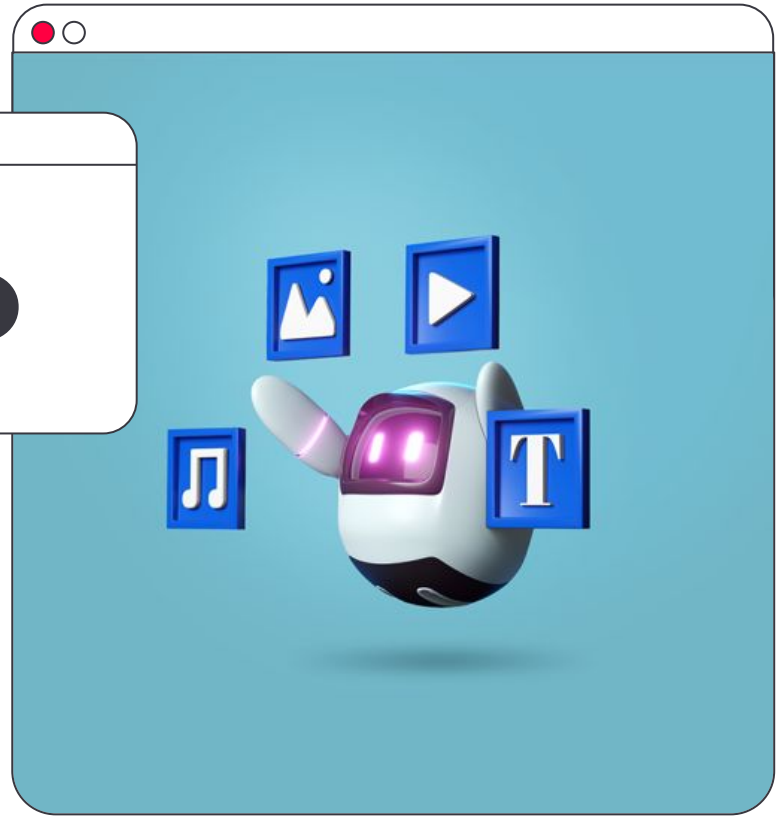
Why a workshop about vector databases?

- Music recommendation
- Semantic search
- Facial recognition
- NLP
- Image search
- RAG (Retrieval Augmented Generation)

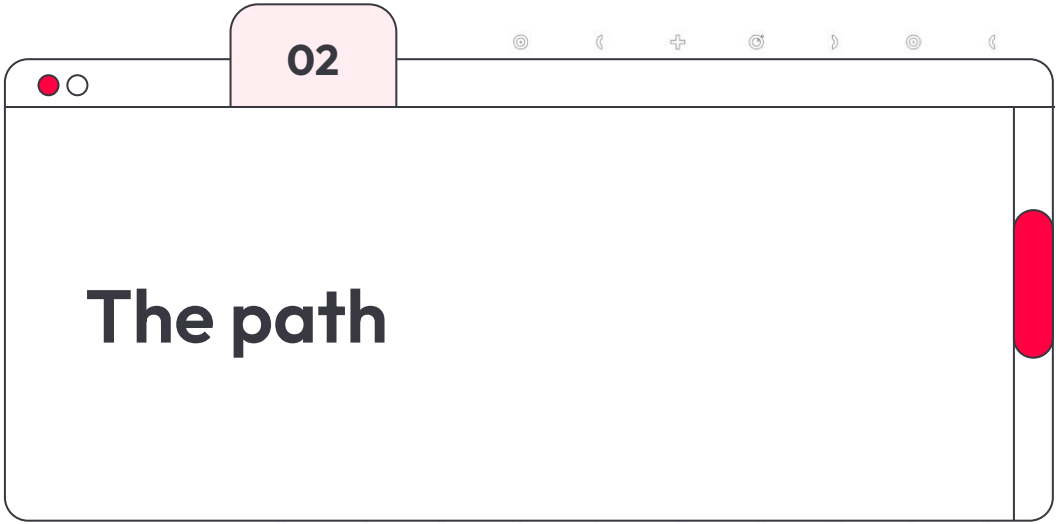



01
Demo

Demo



https://github.com/AurelienMassiot/pycon_lithuania_24_vector_db



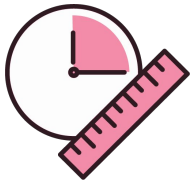
The path

Step by step

01 Code

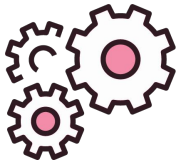
Code an algorithm to search for similar vectors

02 Observe

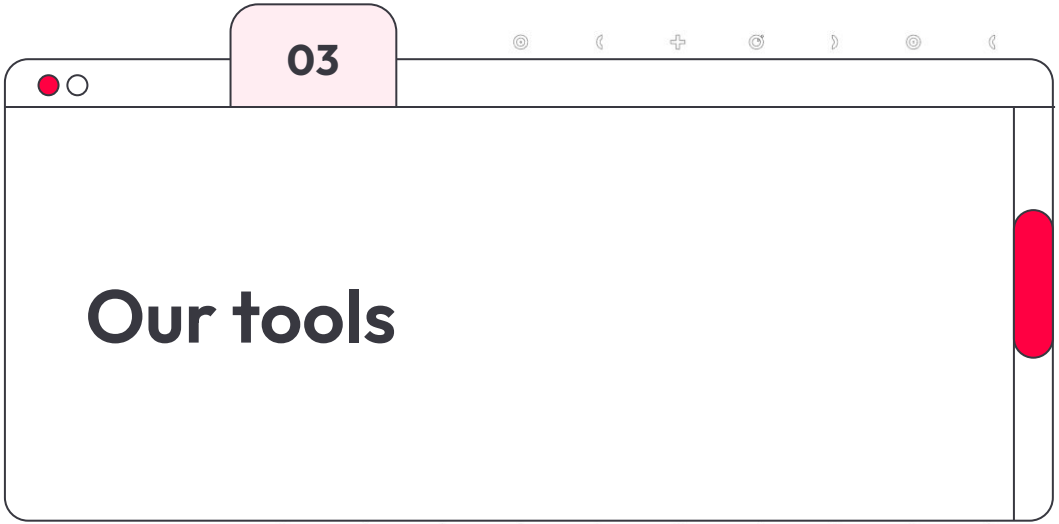


Observe the results in the demo application

03 Improve

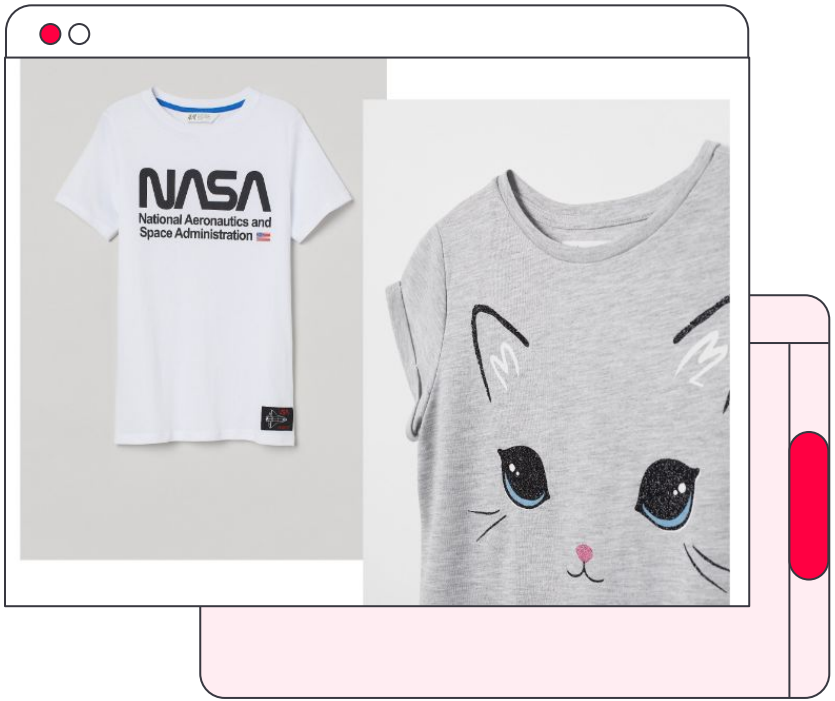


Do it better!



Our tools

Our data



KAGGLE :

H&M Personalized Fashion Recommendations

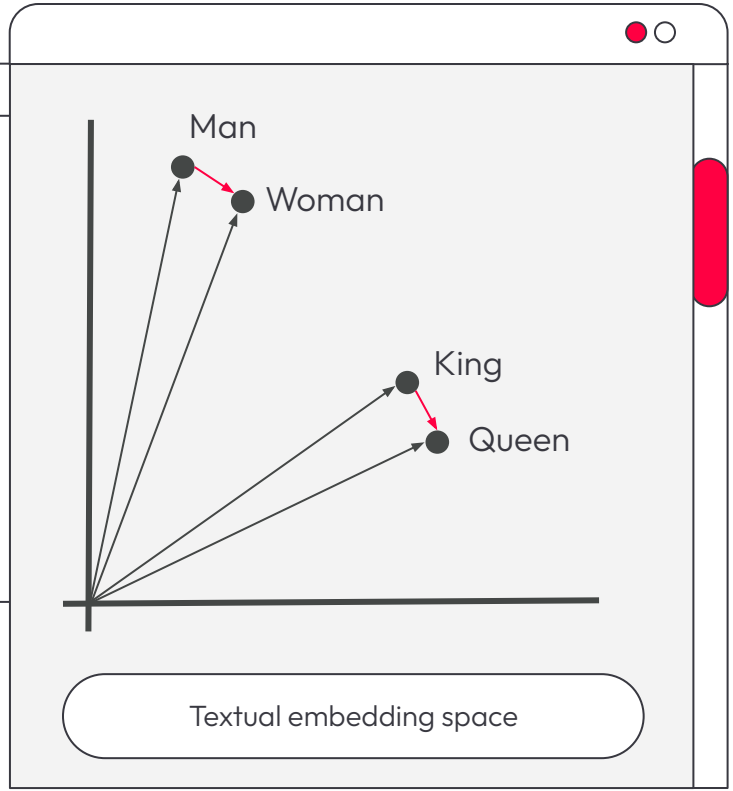
105k images of fashion products from H&M collection

and that's all!

Embeddings

Quick reminder

- Vectors in this workshop are **embeddings**
- This is a vectorial representation of an object
  - Sentence or image in our case
- Captures semantic features in a vector space

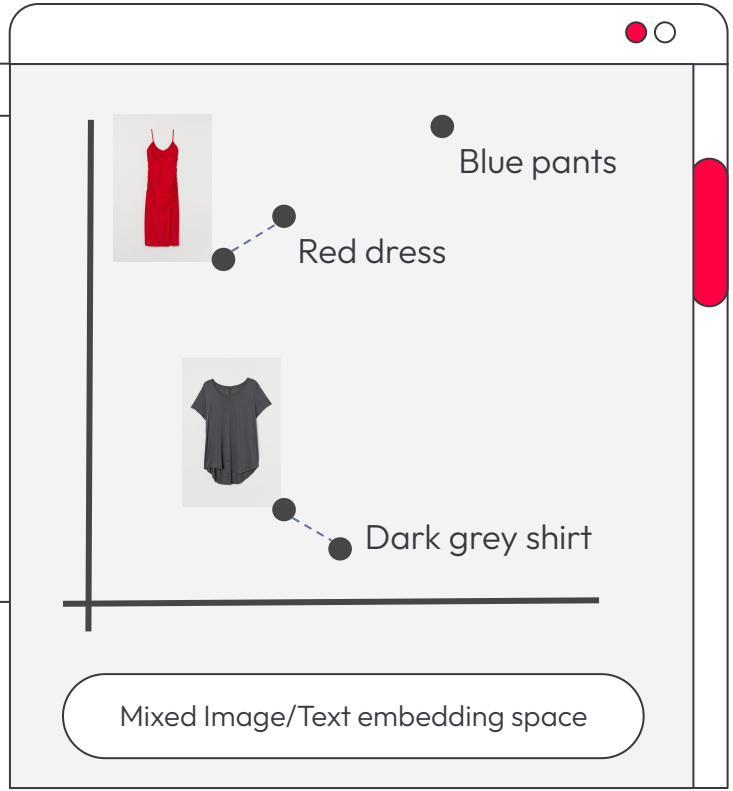


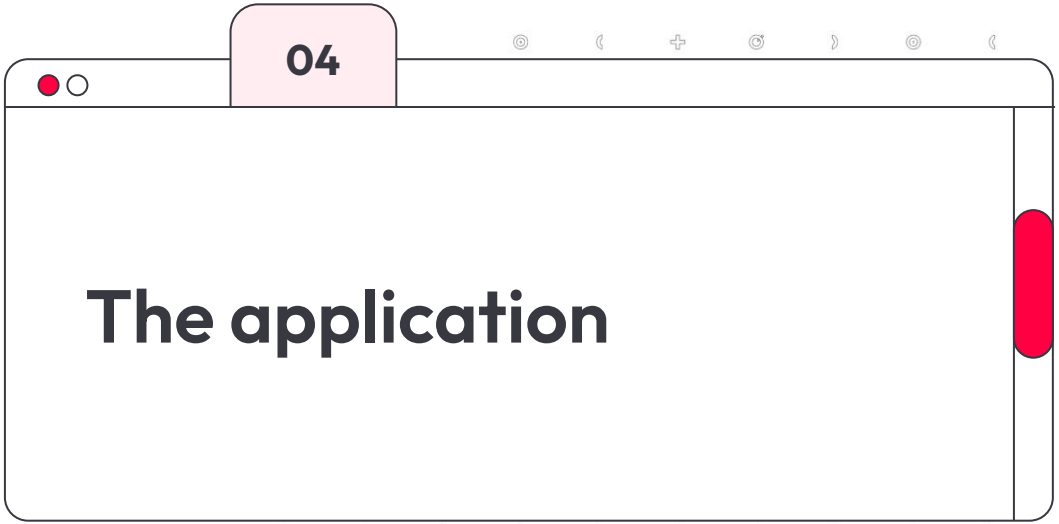
CLIP : Contrastive Language-Image Pre-Training

OpenAI

- Understands and generates visual representations and textual: embeddings
- Trained on 400 million pairs Image - Text description
- A version retrained for fashion: FashionCLIP

Hugging Face





The application

Data flow

105k images embedded with FashionCLIP

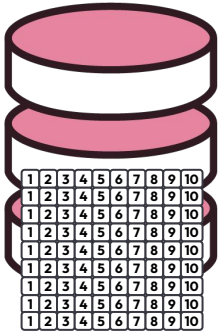
Fashion CLIP

AI

Image to vector

1 2 3 4 5 6 7 8 9 10

1. Insertion in database and indexation



3. Restitution of k nearest vectors

"White tshirt with NASA logo"

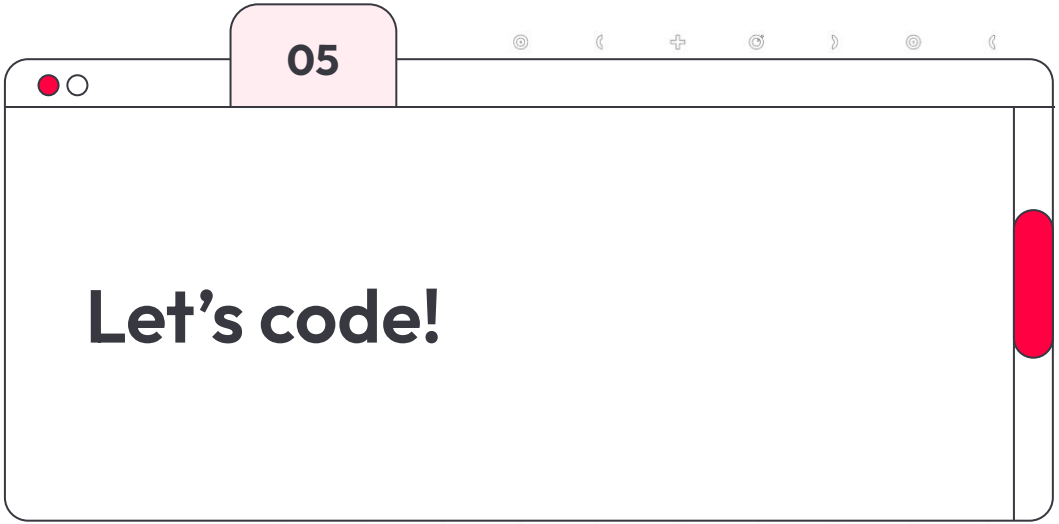
2. Search for k nearest vectors

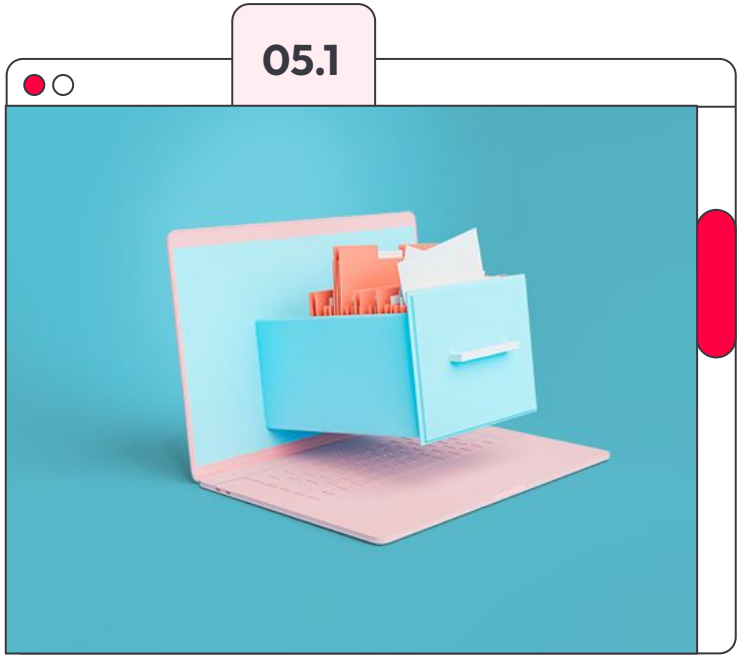
Fashion CLIP

AI

Image to vector

1 2 3 4 5 6 7 8 9 10


05
Let’s code!

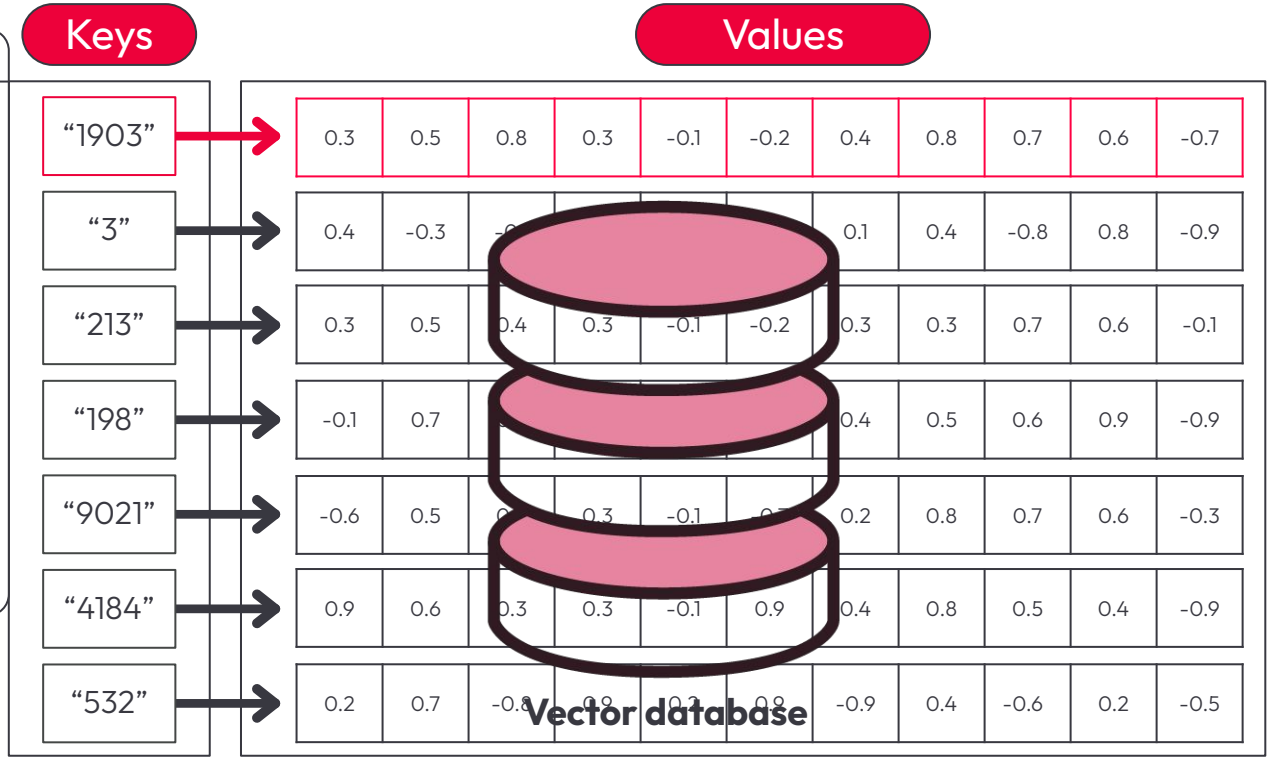


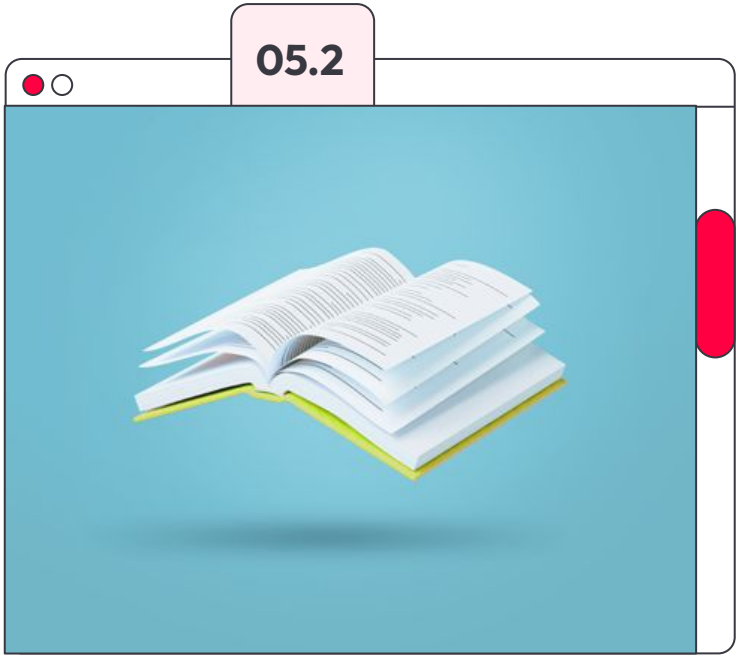
Vectors storage

Insertion of vectors into our database

Python dictionary

- Storing the **105k vectors** in a structured database
- Storing of the **product id** to create the Product > Link vector





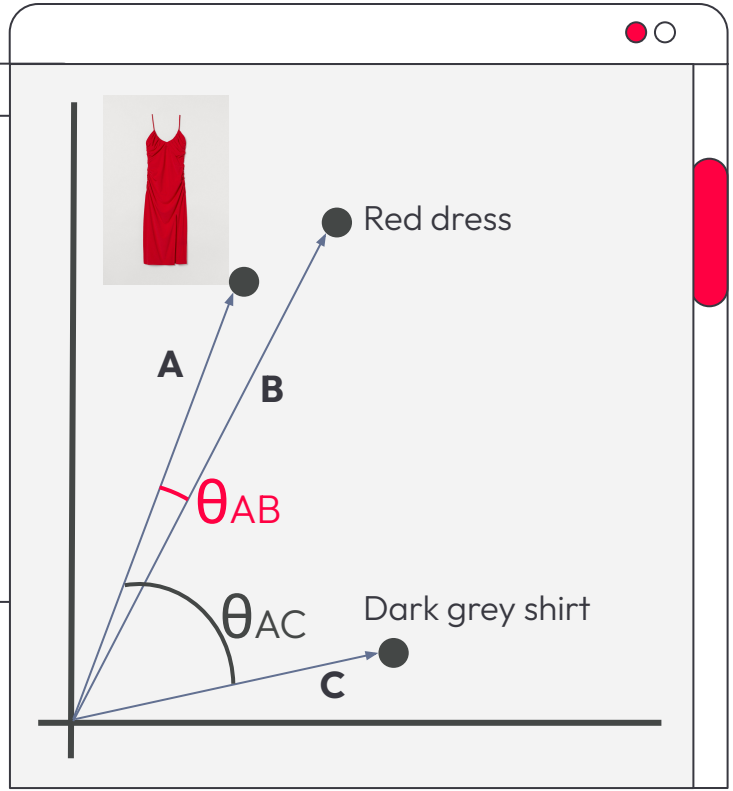
Exhaustive search (or brute-force search)

Cosine similarity

The formula and intuition

- Enables calculating a **similarity** between the vectors based on the **angle** between them

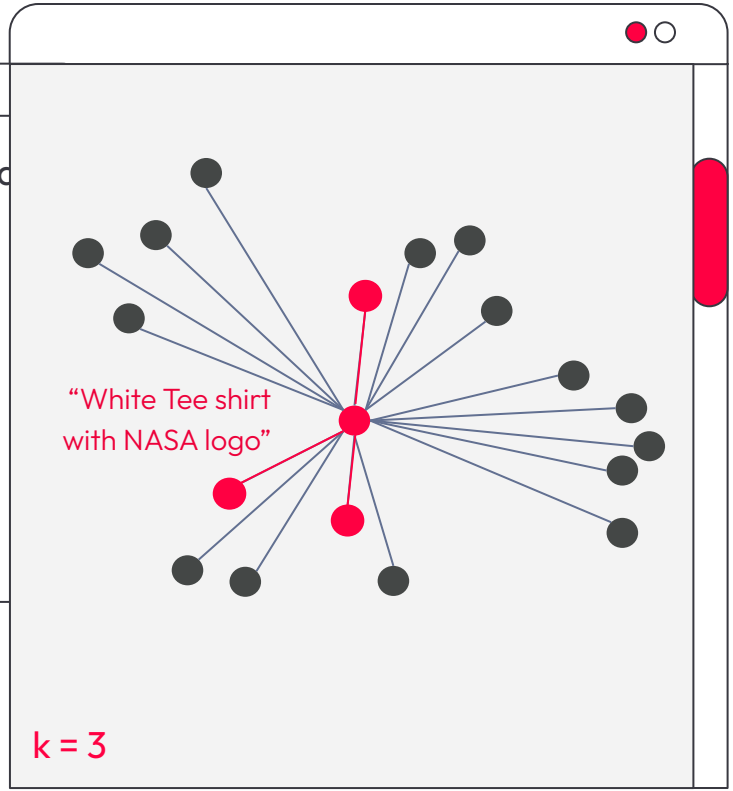
$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|}$$

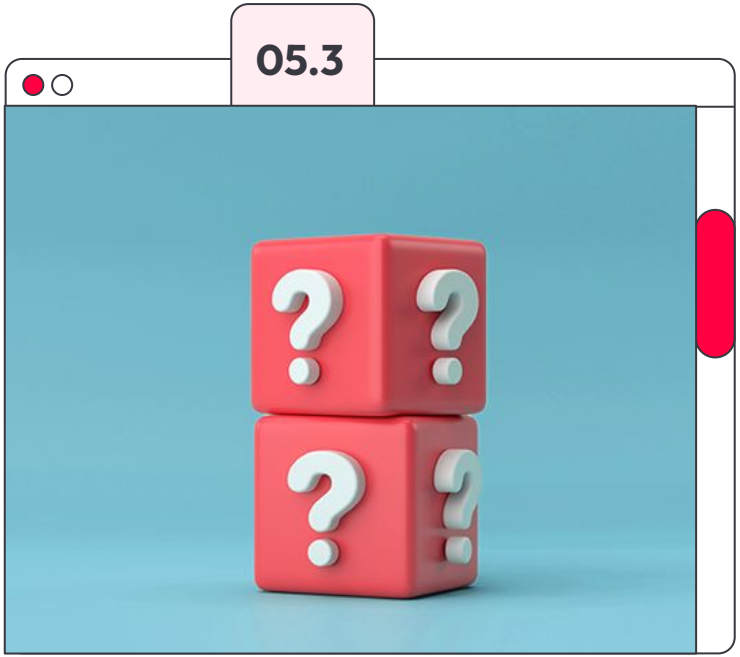


The exhaustive search or brute-force

The steps

- Calculating all the distances between **the request** and the **vector database**
- Sorting the distances and keeping the **k nearest vectors**
- Known as **KNN**



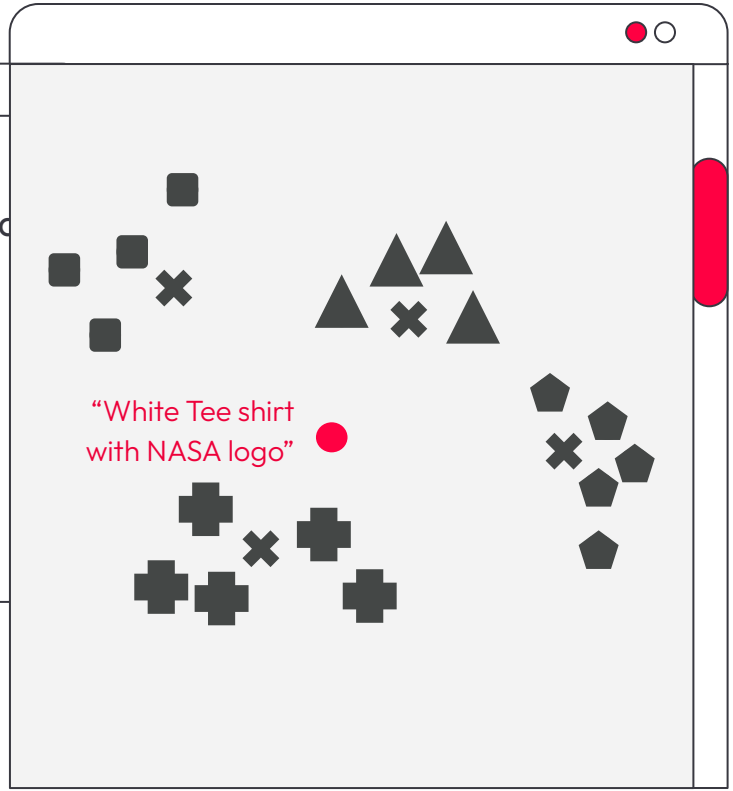


The inverted index

K-means indexing

The steps

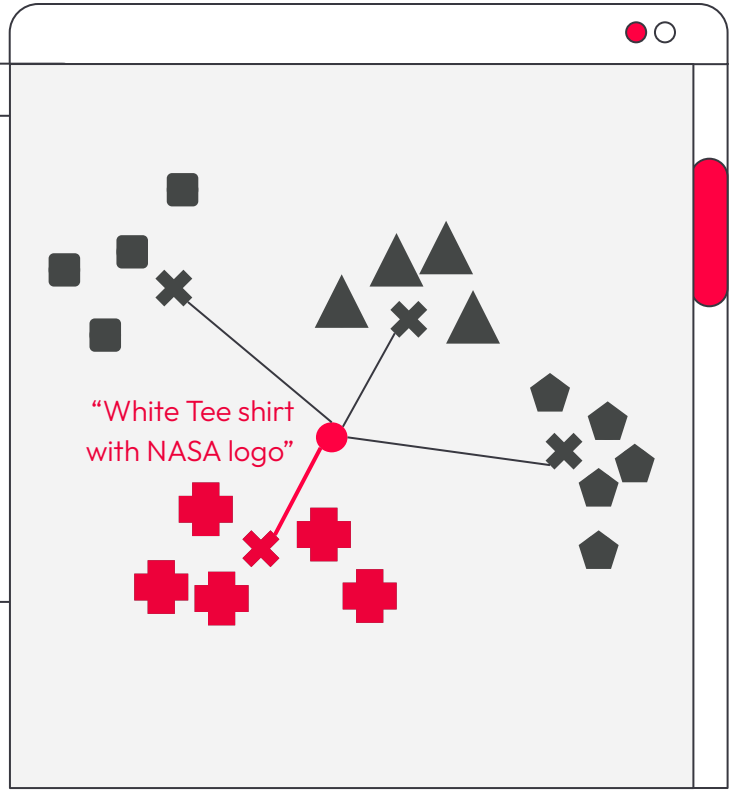
- Segment the database with **K-means** and assign ec
  vector to a cluster :
  - The index
- Saving the **centroid** coordinates
  - The codebook ✖



The inverted index search

The steps

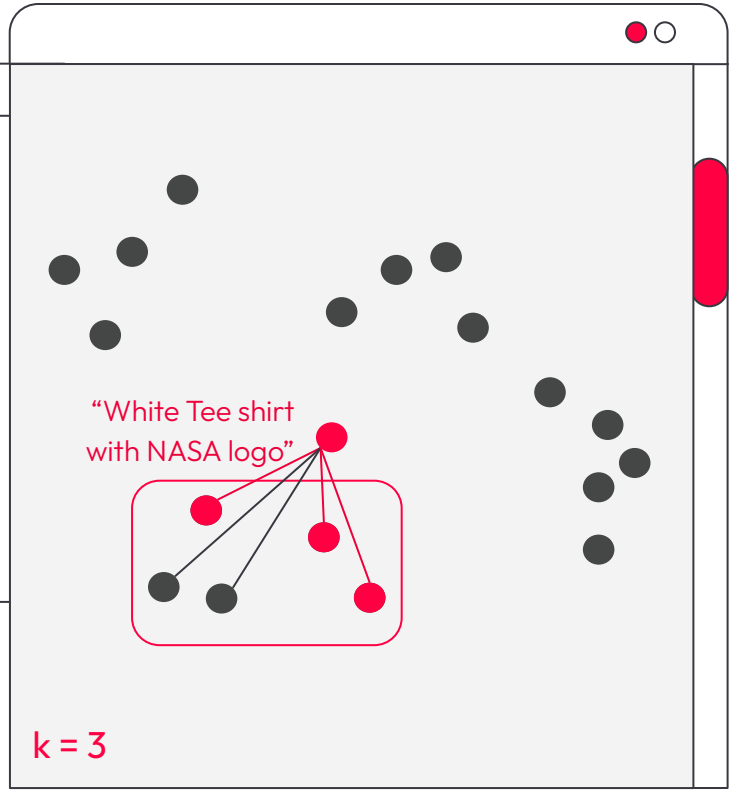
- Calculating the distances between the request and the centroids **(the codebook)**
- Select the cluster whose **centroid** is **the closest**



The inverted index search

The steps

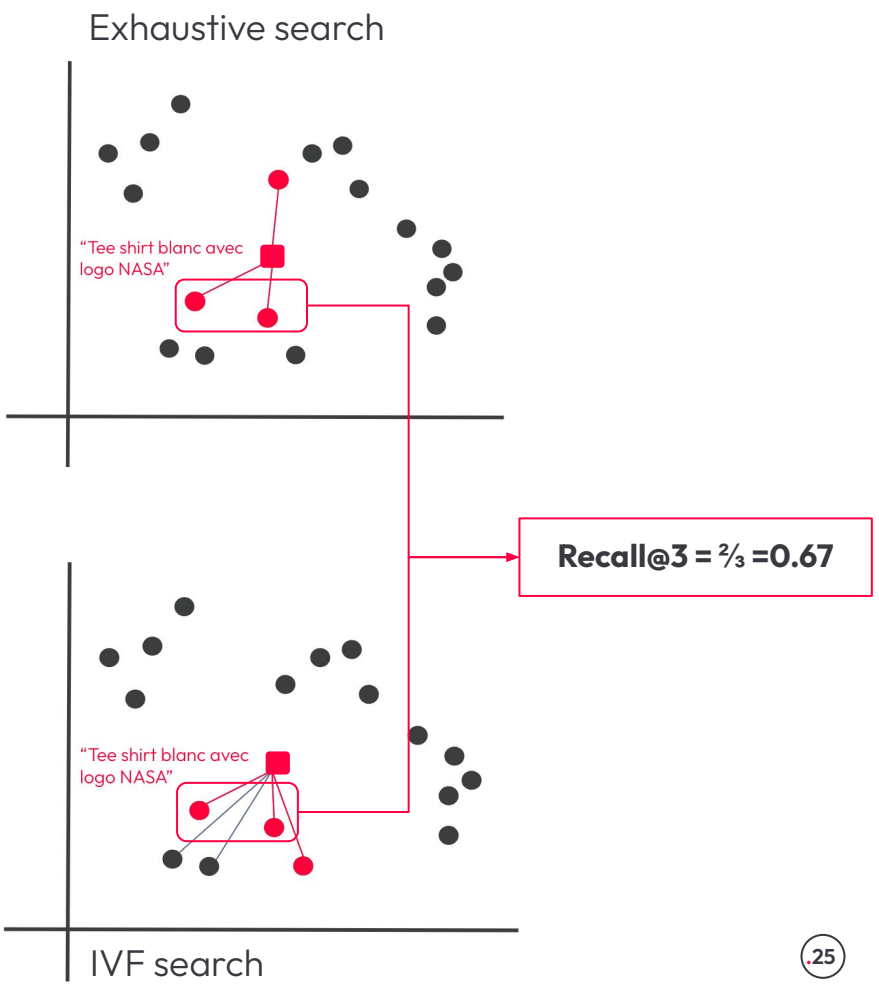
- Perform an exhaustive research vectors belonging to the **selected cluster**



Evaluate the differences

Calculation reminder

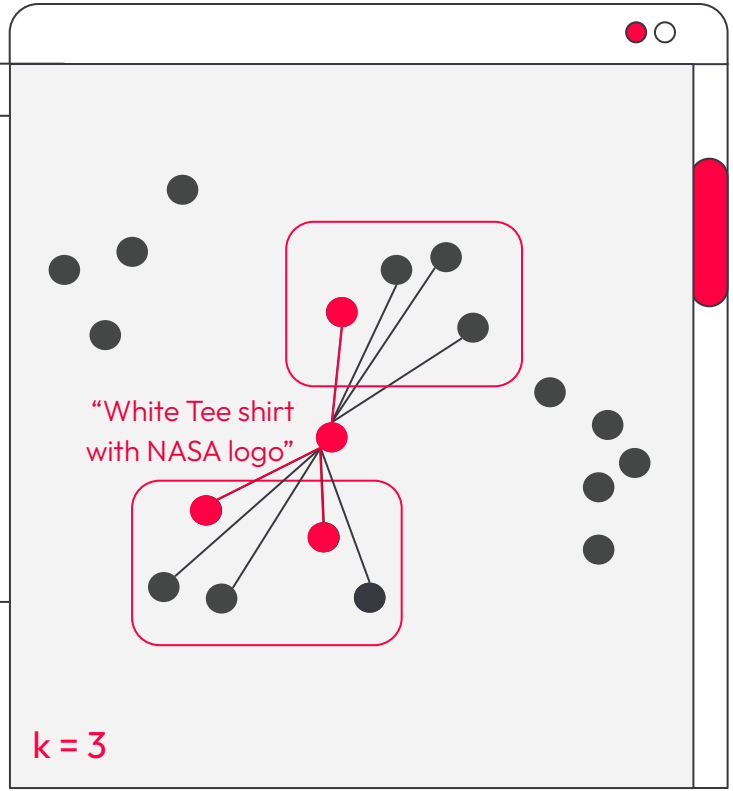
- The **“exact” search** is the exhaustive search

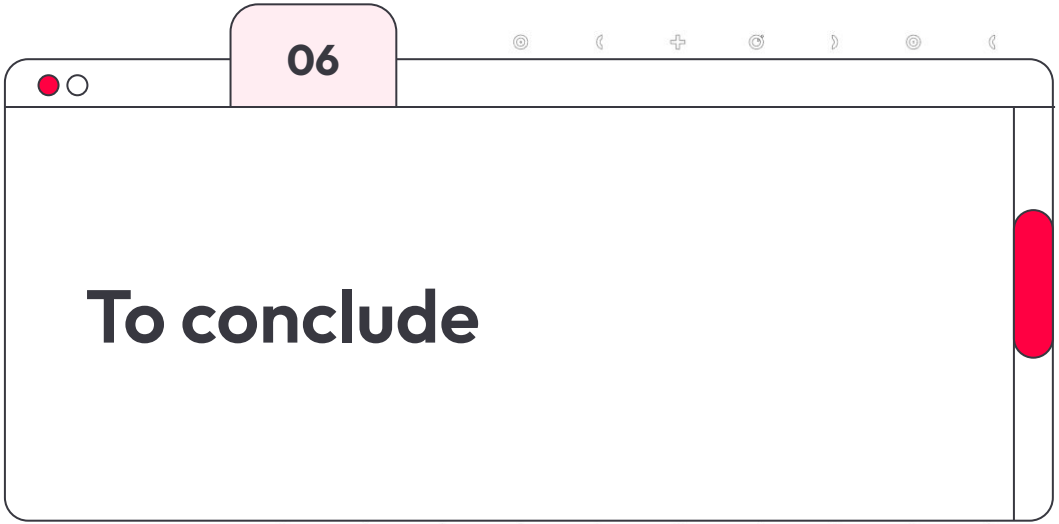


The inverted index search and several probes

The steps

- Perform an exhaustive search on vectors belonging to the **N closest clusters**





To conclude

TAKE AWAYS

It's a breeze

Building a multimodal search engine is easy with the emergence of models like CLIP and vector databases

Very prevalent algorithms

IVF, HNSW, LSH, ANNOY available on the marketplace databases

Dilemma precision/performance

Your database index and parameters are key for better precision and performance:
https://ann-benchmarks.com/

Benchmarks

https://ann-benchmarks.com/

Benchmarking Results

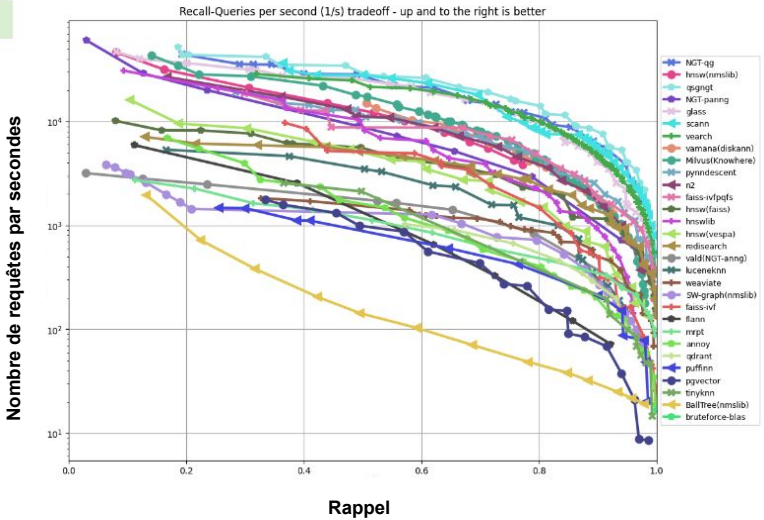
Results are split by distance measure and dataset. In the bottom, you can find an overview of an algorithm's performance on all datasets. Each dataset is annoted by *(k = ...)*, the number of nearest neighbors an algorithm was supposed to return. The plot shown depicts *Recall* (the fraction of true nearest neighbors found, on average over all queries) against *Queries per second*. Clicking on a plot reveils detailled interactive plots, including approximate recall, index size, and build time.

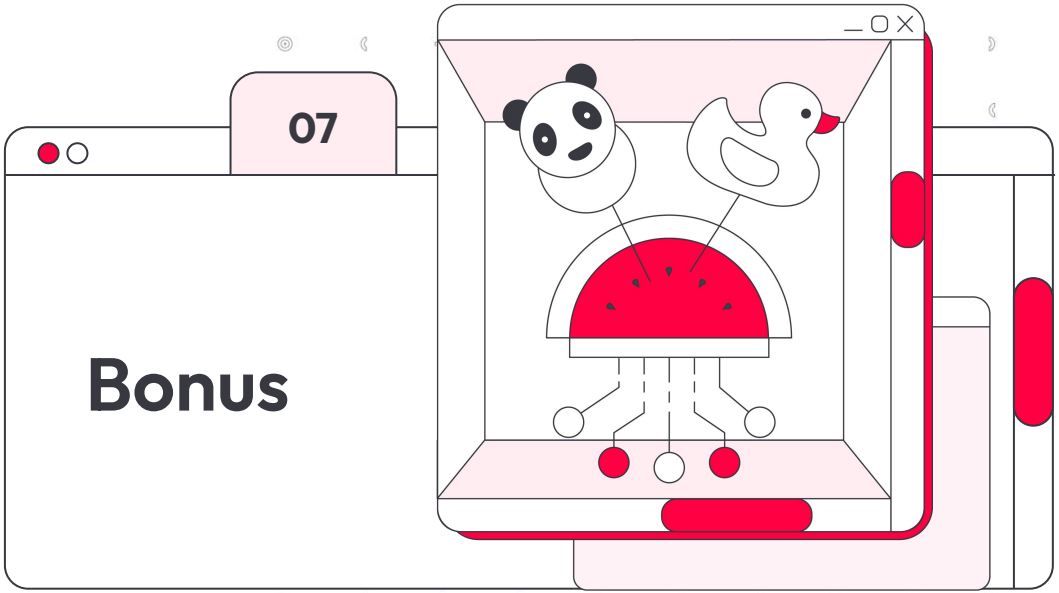
Benchmarks for Single Queries

Results by Dataset

Distance: Angular

glove-100-angular (k = 10)




07
Bonus

Stay in touch

https://github.com/AurelienMassiot/pycon_lithuania_24_vector_db

Aurélien Massiot

Head of Data User Centric & Senior ML Engineer @ OCTO Technology