



La **météorite GPT** annonce t'elle la disparition  
des **dinosaures "moteur de recherche"** ?



THE DARK  
CHAPTER OF THE  
DEVFEST  
NANTES  
2024

openfeedback



The  
**searchengine**rosaur



openfeedback





# Elasticsearch

*Lucene*

[openfeedback](#)







[openfeedback](#)





66

These are not the droids  
you are looking for.

[openfeedback](#)



```
GET /_analyze
{
  "char_filter": [ "html_strip" ],
  "tokenizer": "standard",
  "filter": [ "lowercase", "stop", "snowball" ],
  "text": "These are <em>not</em> the droids
          you are looking for."
}
```

[openfeedback](#)



```
"char_filter": "html_strip"
```

These are `<em>not</em>` the droids you are looking for.



These are not the droids you are looking for.

[openfeedback](#)





```
"tokenizer": "standard"
```

These are not the droids you are looking for.



```
These  
are  
not  
the  
droids  
you  
are  
looking  
for
```

[openfeedback](#)



"filter": "lowercase"

**T**hese  
are  
not  
the  
droids  
you  
are  
looking  
for



**t**hese  
are  
not  
the  
droids  
you  
are  
looking  
for

[openfeedback](#)



```
"filter": "stop"
```

**These  
are  
not  
the**

droids  
you  
**are**  
looking  
**for**



droids  
you  
  
looking

[openfeedback](#)





```
"filter": "snowball"
```

droids	→	droid
you		you
looking		look

[openfeedback](#)



These are `<em>not</em>` the **droids you** are **looking** for.

```
{ "tokens": [{
  "token": "droid",
  "start_offset": 27, "end_offset": 33,
  "type": "<ALPHANUM>", "position": 4
}, {
  "token": "you",
  "start_offset": 34, "end_offset": 37,
  "type": "<ALPHANUM>", "position": 5
}, {
  "token": "look",
  "start_offset": 42, "end_offset": 49,
  "type": "<ALPHANUM>", "position": 7
}]]
```

[openfeedback](#)





**Semantic**  
search  
≠  
**Literal**  
matches

similarweb

**YOU'RE COMPARING  
APPLES TO NECTARINES**



openfeedback







2016



2018



**David Pilato**  
@dadoonet

openfeedback



# Elasticsearch

You Know, for Search

[openfeedback](#)



# Elasticsearch

You Know, for **Vector** Search

[openfeedback](#)





# What is a **Vector**?





openfeedback



# Embeddings represent your data

## Example: 1-dimensional vector

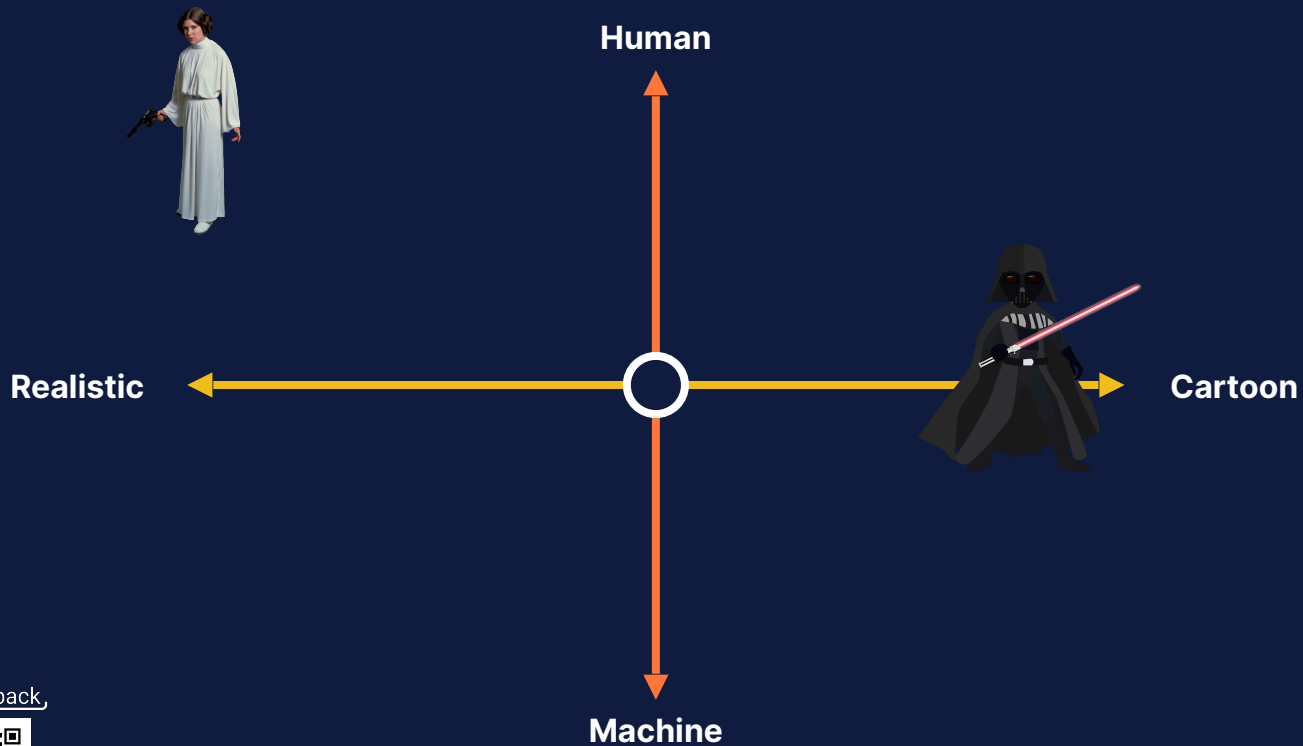




Character	Vector
	$[-1]$
	$[1]$

[openfeedback](#)



# Multiple dimensions represent different data aspects

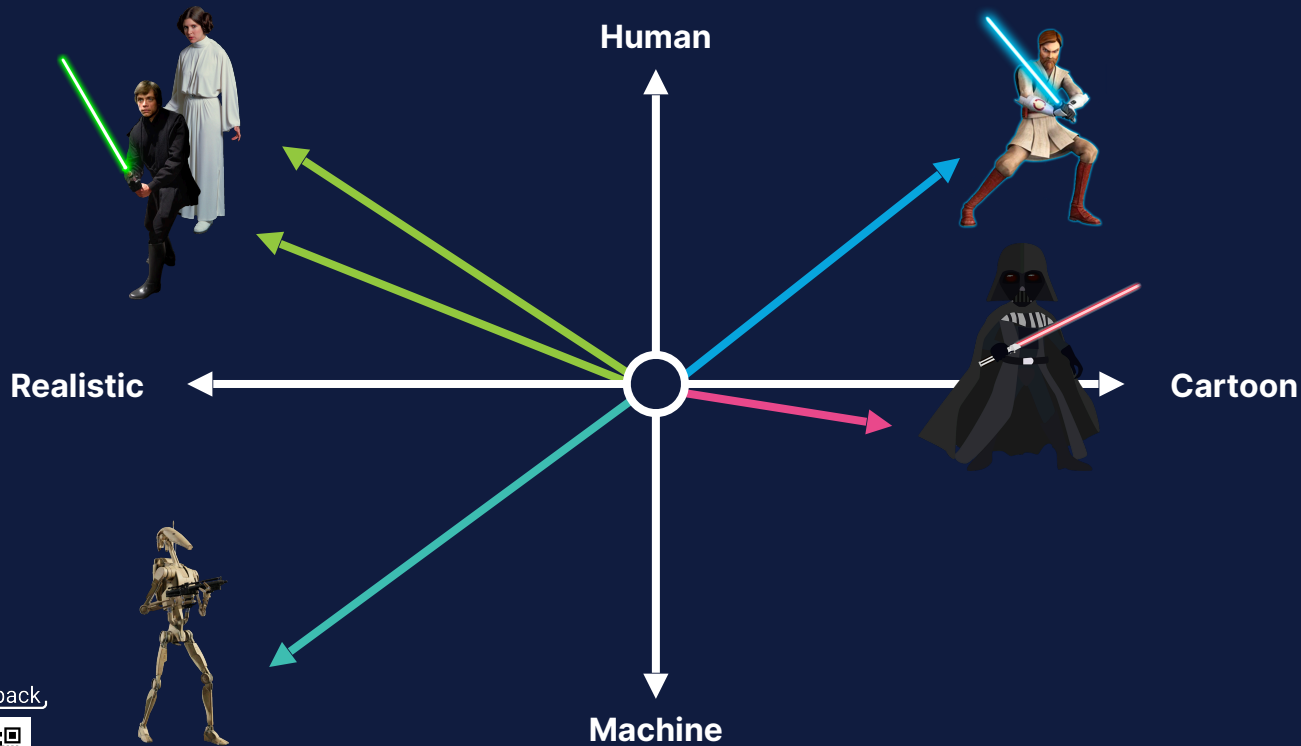







Character	Vector
	$[-1, 1]$
	$[1, 0]$

openfeedback



# Similar data is grouped together

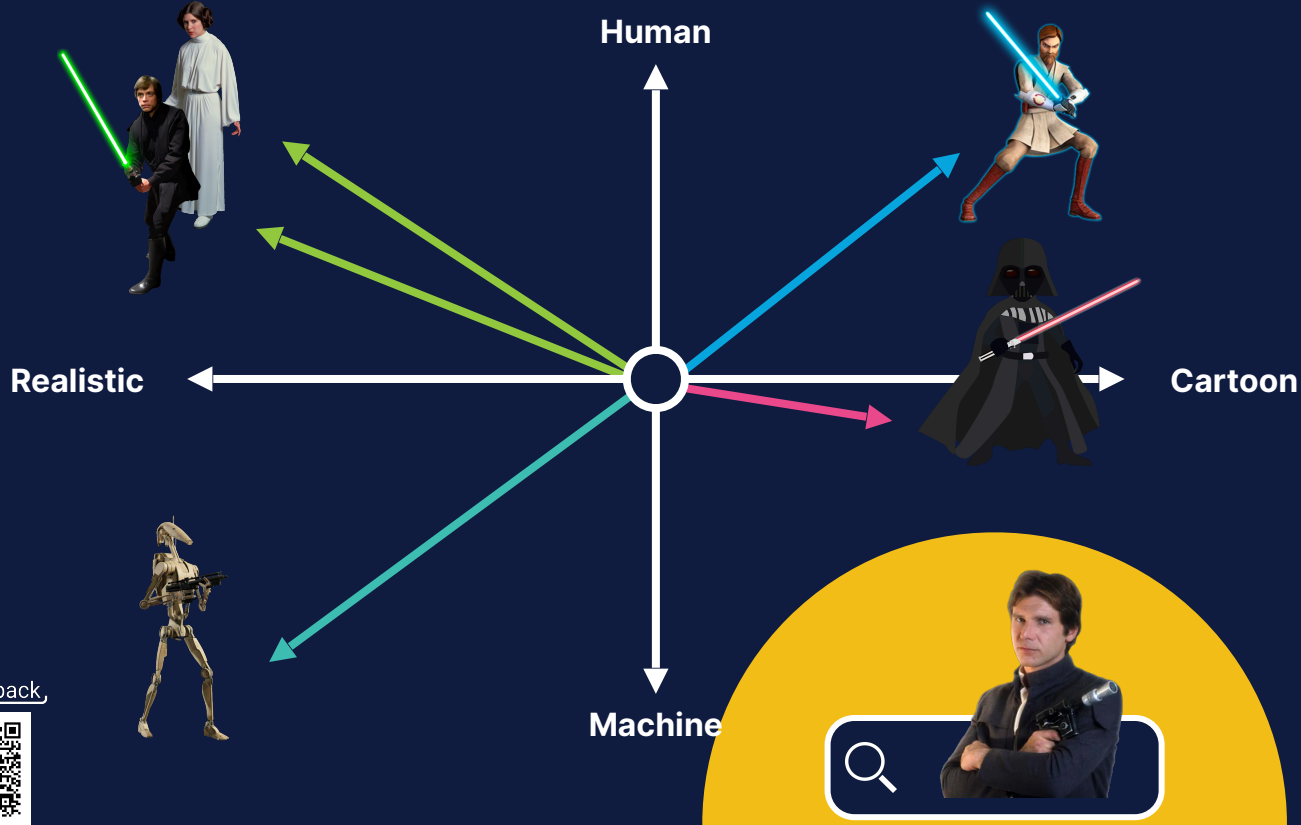


Character	Vector
	$[-1.0, 1.0]$
	$[1.0, 0.0]$
	$[-1.0, 0.8]$
	$[1.0, 1.0]$
	$[-1.0, -1.0]$

openfeedback



# Vector search ranks objects by similarity (~relevance) to the query



Rank	Result
Query	
1	
2	
3	
4	
5	

openfeedback



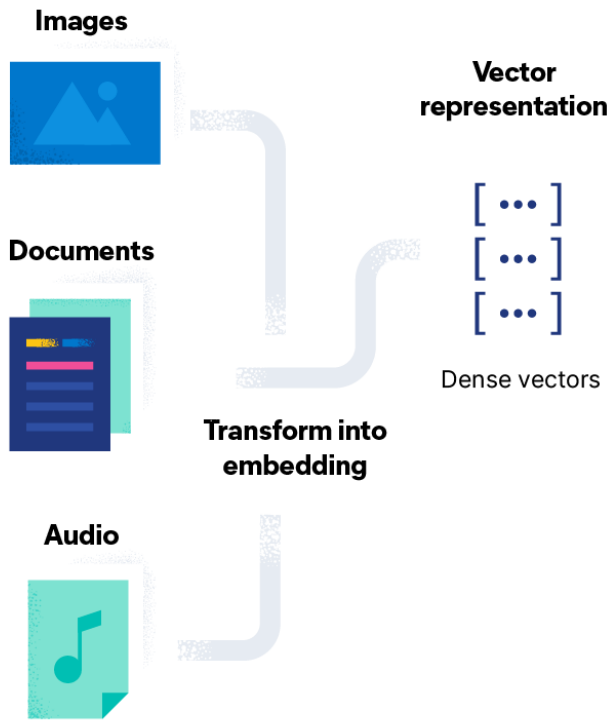


# How do you index **vectors**?

[openfeedback](#)



# Architecture of Vector Search



# dense\_vector field type

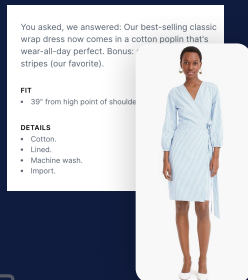
```
PUT ecommerce
{
  "mappings": {
    "properties": {
      "description": {
        "type": "text"
      }
      "desc_embedding": {
        "type": "dense_vector"
      }
    }
  }
}
```

[openfeedback](#)



# Data Ingestion and Embedding Generation

POST ecommerce/\_doc



 Source data



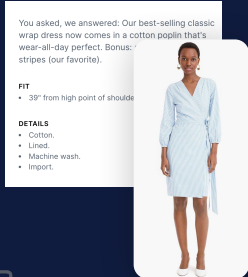
POST /ecommerce/\_doc

```
{
  "_id": "product-1234",
  "product_name": "Summer Dress",
  "description": "Our best-selling...",
  "Price": 118,
  "color": "blue",
  "fabric": "cotton",
  "desc_embedding": [0.452, 0.3242, ...],
  "img_embedding": [0.012, 0.0, ...]
}
```

openfeedback



# With Elastic ML



```
{
  "_id": "product-1234",
  "product_name": "Summer Dress",
  "description": "Our best-selling...",
  "Price": 118,
  "color": "blue",
  "fabric": "cotton",
  "desc_embedding": [0.452, 0.3242, ...]
}
```

## Source data

```
{
  "_id": "product-1234",
  "product_name": "Summer Dress",
  "description": "Our best-selling...",
  "Price": 118,
  "color": "blue",
  "fabric": "cotton",
}
```

POST /ecommerce/\_doc

### ML Inference pipelines

Inference pipelines will be run as processors from the Enterprise Search Ingest Pipeline

- ml-inference-embedding-generation**
  - Deployed | pytorch | text\_embedding
- ml-inference-emotional-analysis**
  - Deployed | pytorch | text\_classification

Learn more about deploying ML models in Elastic

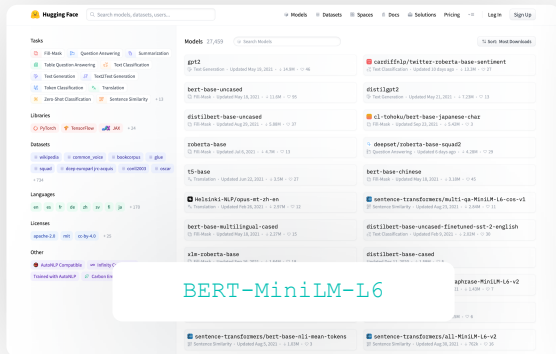


openfeedback

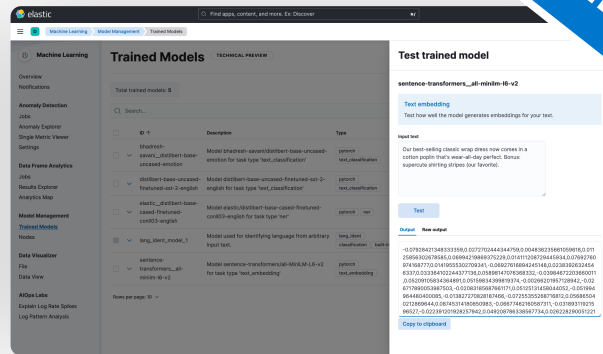




# Eland Imports PyTorch Models



```
$ eland import hub_model
--url https://cluster_URL --hub-
model-id BERT-MiniLM-L6 --task-
type text_embedding --start
```



PyTorch



Load it



Manage models

Select the appropriate model

# Elastic's range of supported NLP models

- **Fill mask model**

Mask some of the words in a sentence and predict words that replace masks

- **Named entity recognition model**

NLP method that extracts information from text

- **Text embedding model**

Represent individual words as numerical vectors in a predefined vector space

- **Text classification model**

Assign a set of predefined categories to open-ended text

- **Question answering model**

Model that can answer questions given some or no context

- **Zero-shot text classification model**

Model trained on a set of labeled examples, that is able to classify previously unseen examples

## Third party fill-mask models

- BE
- Dis
- MP
- Ro

## Third party text classification models

- BE
- Dis
- MP
- Ro

## Third party named entity recognition models

- BE
- Dis
- MP
- Ro

## Third party question answering models

- BE
- Dis
- MP
- Ro

## Third party text embedding models

- BE
- Dis
- MP
- Ro

## Third party zero-shot text classification models

- BART large mnli
- DistilBERT base model (uncased)
- **DistilBart MNLI**
- MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices
- NLI DistilRoBERTa base
- NLI RoBERTa base
- SqueezeBERT

openfeedback



Full list at: [ela.st/nlp-supported-models](https://ela.st/nlp-supported-models)

# How do you search **vectors**?

[openfeedback](#)



# Architecture of Vector Search



# knn query

 ✕ 

```
GET ecommerce/_search
{
  "query" : {
    "bool" : {
      "must" : [{
        "knn" : {
          "field" : "desc_embedding",
          "num_candidates" : 50,
          "query_vector" : [0.123, 0.244, ...]
        }
      }
    ],
    "filter" : {
      "term" : {
        "department" : "women"
      }
    }
  },
  "size" : 10
}
```

openfeedback





# knn query (with Elastic ML)

 x 

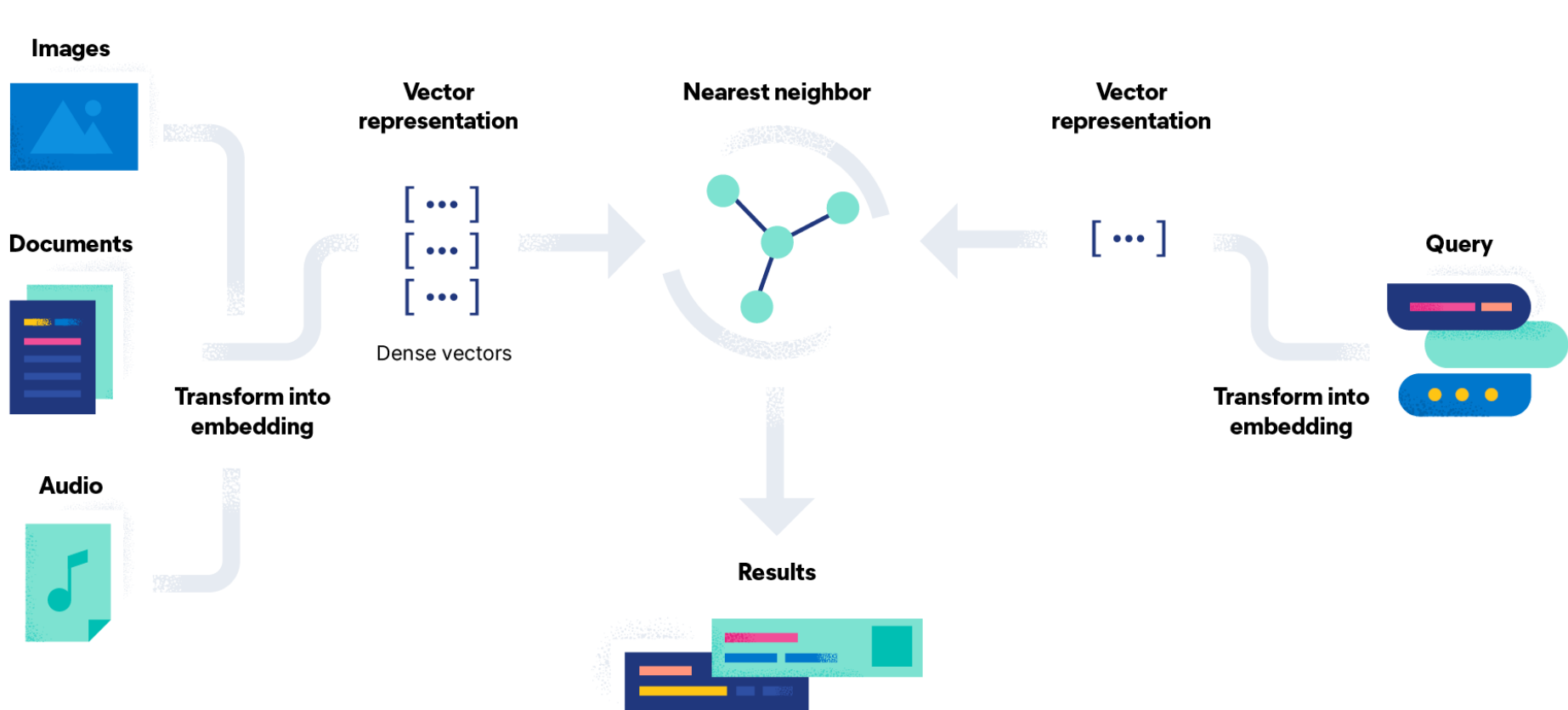
Transformer model

```
GET product-catalog/_search
{
  "query" : {
    "bool" : {
      "must" : [{
        "knn" : {
          "field" : "desc_embedding",
          "num_candidates" : 50,
          "query_vector_builder" : {
            "text_embedding" : {
              "model_text" : "summer clothes",
              "model_id" : <text-embedding-model>
            }
          }
        }
      ]
    }
  },
  "filter" : {
    "term" : {
      "department" : "women"
    }
  }
},
  "size" : 10
}
```

[openfeedback](#)



# Architecture of Vector Search



# Choice of Embedding Model

## Start with Off-the Shelf Models

- Text data: Hugging Face (like Microsoft's E5)
- Images: OpenAI's CLIP

## Extend to Higher Relevance

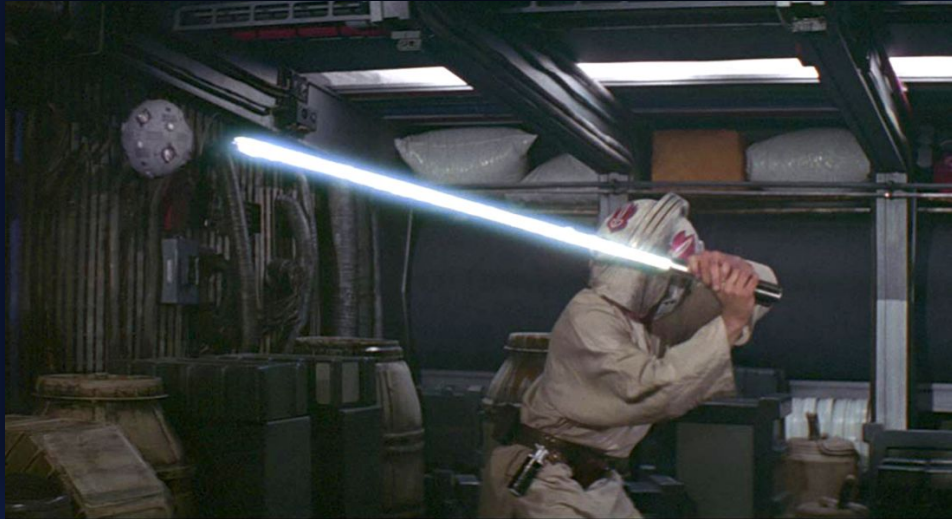
- Apply hybrid scoring
- Bring Your Own Model: requires expertise + labeled data

[openfeedback](#)



# Problem

training vs actual use-case



[openfeedback](#)





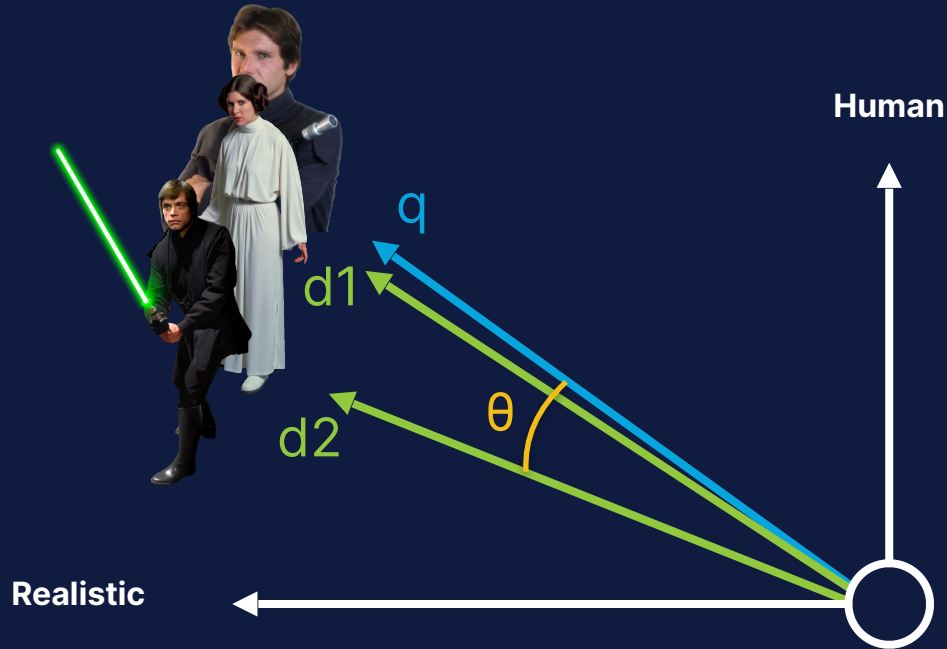
**But how does it**  
really work?



[openfeedback](#)



# Similarity



$$\cos(\theta) = \frac{\vec{q} \cdot \vec{d}}{|\vec{q}| \times |\vec{d}|}$$

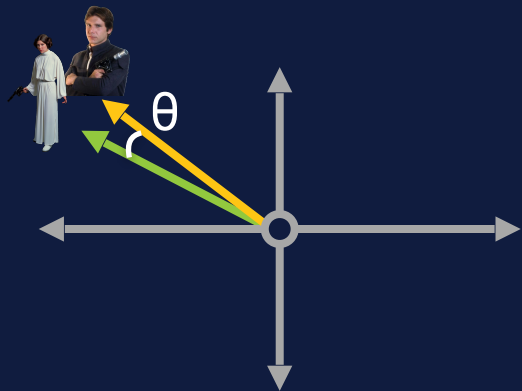
$$\text{\_score} = \frac{1 + \cos(\theta)}{2}$$

[openfeedback](#)





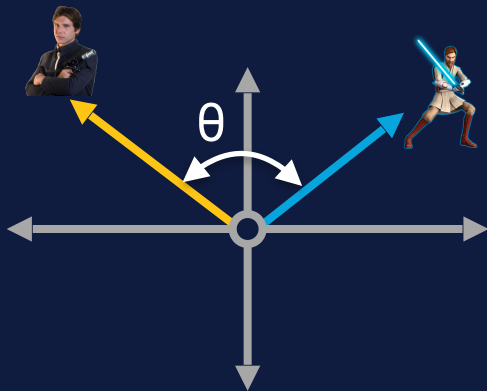
# Similarity: cosine (cosine)



## Similar vectors

$\theta$  close to 0  
 $\cos(\theta)$  close to **1**

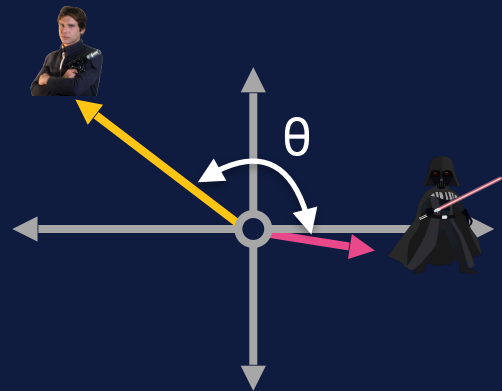
$$\text{\_score} = \frac{1 + 1}{2} = 1$$



## Orthogonal vectors

$\theta$  close to  $90^\circ$   
 $\cos(\theta)$  close to **0**

$$\text{\_score} = \frac{1 + 0}{2} = 0.5$$



## Opposite vectors

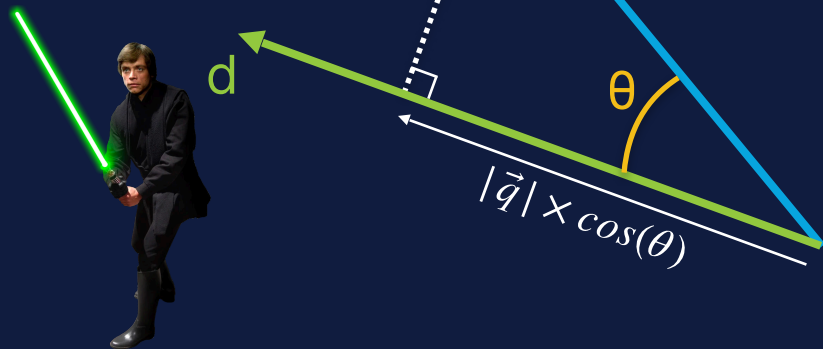
$\theta$  close to  $180^\circ$   
 $\cos(\theta)$  close to **-1**

$$\text{\_score} = \frac{1 - 1}{2} = 0$$

[openfeedback](#)



# Similarity: Dot Product (`dot_product` or `max_inner_product`)



$$\vec{q} \times \vec{d} = |\vec{q}| \times \cos(\theta) \times |\vec{d}|$$

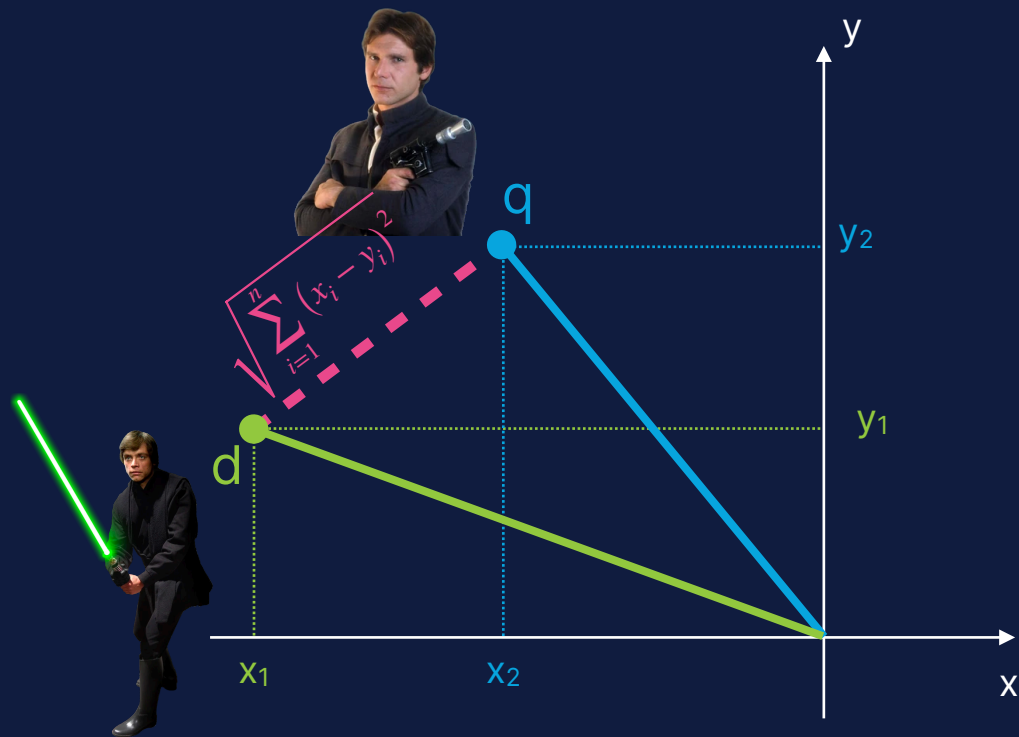
$$\text{\_score}_{float} = \frac{1 + \text{dot\_product}(q, d)}{2}$$

$$\text{\_score}_{byte} = \frac{0.5 + \text{dot\_product}(q, d)}{32768 \times \text{dims}}$$

[openfeedback](#)



# Similarity: Euclidean distance ( $l2\_norm$ )



$$l2\_norm_{q,d} = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$
$$\_score = \frac{1}{1 + (l2\_norm_{q,d})^2}$$

[openfeedback](#)



# Brute Force



openfeedback



# Hierarchical Navigable Small Worlds (HNSW)

One popular approach



**HNSW:** a layered approach that simplifies access to the nearest neighbor



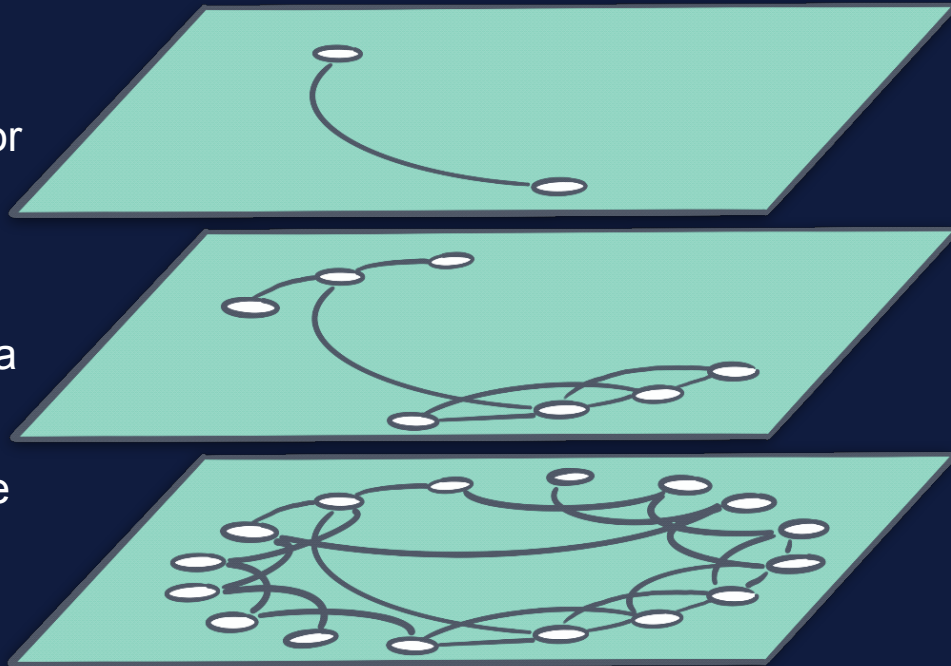
**Tiered:** from coarse to fine approximation over a few steps



**Balance:** Bartering a little accuracy for a lot of scalability



**Speed:** Excellent query latency on large scale indices



openfeedback



# Scaling Vector Search

## Vector search

1. Needs lots of memory
2. Indexing is slower
3. Merging is slow

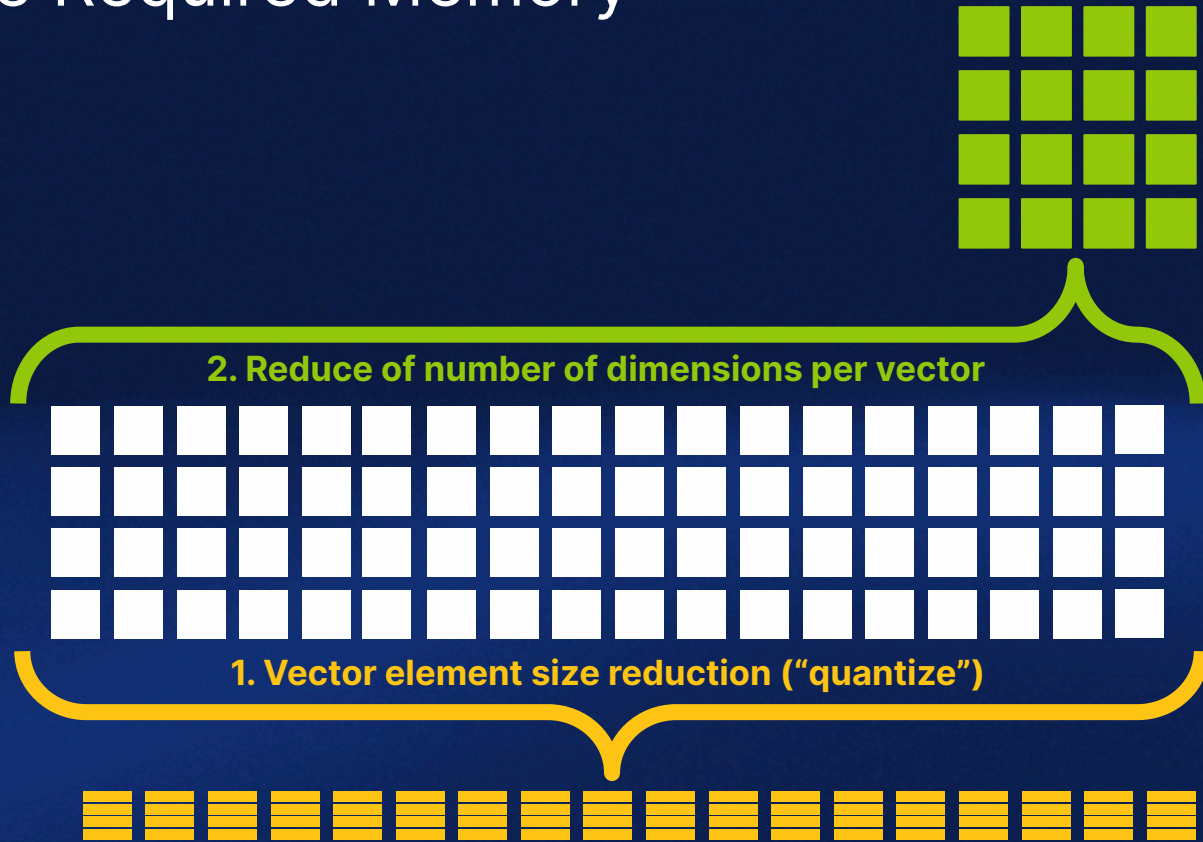
\* Continuous improvements in Lucene + Elasticsearch

## Best practices

1. Avoid searches during indexing
2. Exclude vectors from `_source`
3. Reduce vector dimensionality
4. Use byte rather than float



# Reduce Required Memory

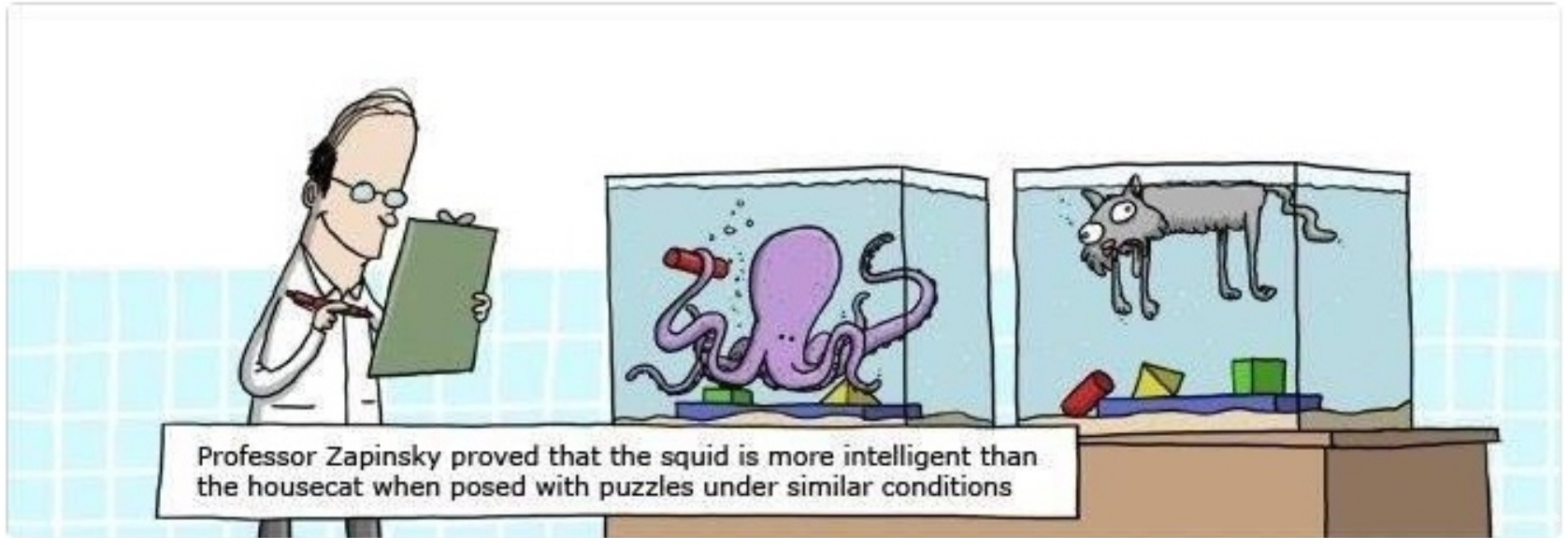


openfeedback





# Benchmarking





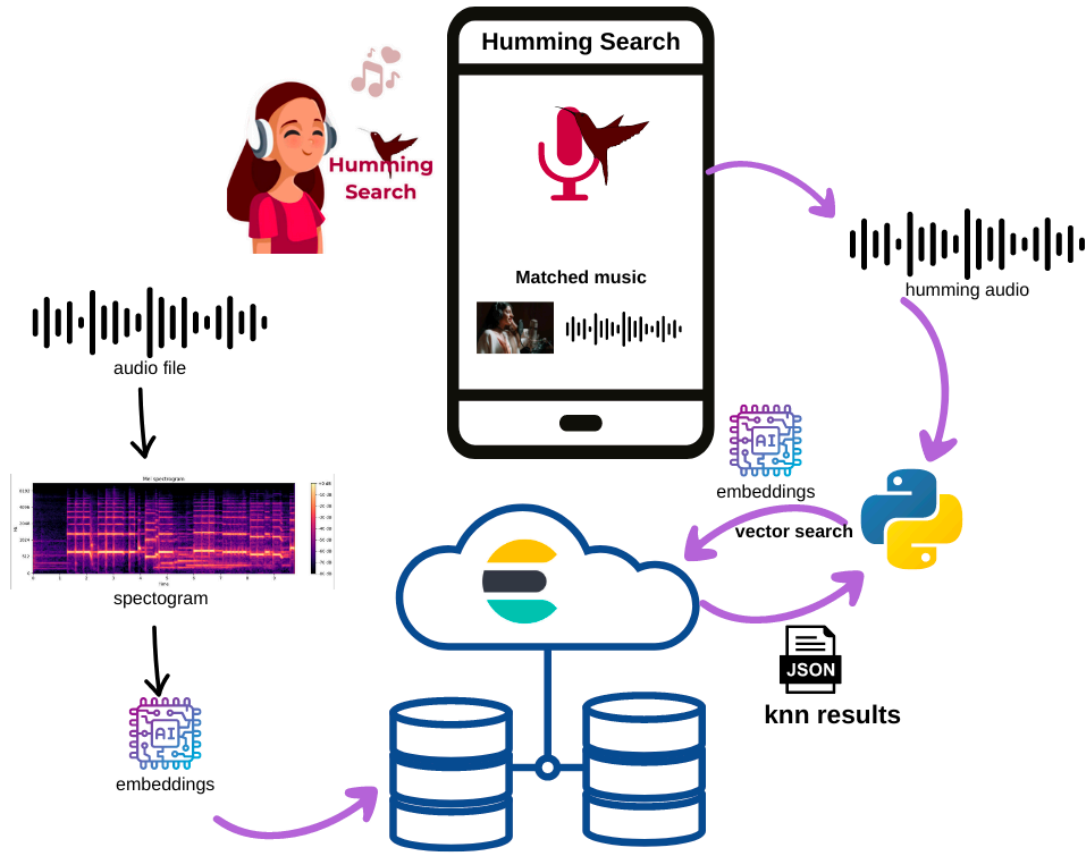
<https://djdadoo.pilato.fr/>

openfeedback



16/09/2023





<https://github.com/dadoonet/music-search/>

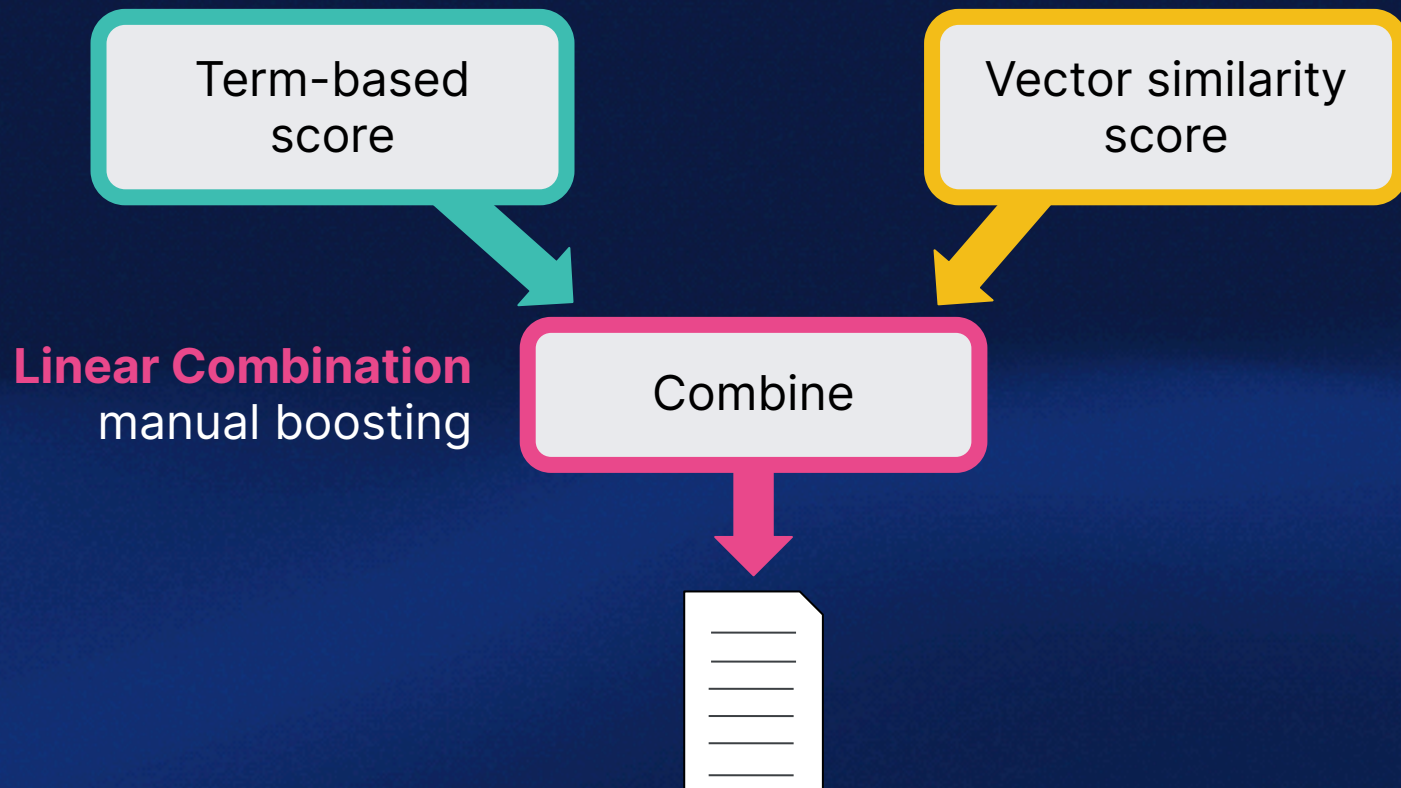
# Elasticsearch

You Know, for **Hybrid** Search

[openfeedback](#)



# Hybrid scoring



[openfeedback](#)



GET ecommerce/\_search

```
{
  "query" : {
    "bool" : {
      "must" : [{
        "match" : {
          "description" : {
            "query": "summer clothes",
            "boost": 0.1
          }
        }
      ]
    }, {
      "knn" : {
        "field": "desc_embedding",
        "query_vector": [0.123, 0.244, ...],
        "num_candidates": 50,
        "boost": 2.0,
        "filter" : {
          "term" : {
            "department": "women"
          }
        }
      }
    }
  ],
  "filter" : {
    "range" : { "price": { "lte": 30 } }
  }
}
```

summer clothes

pre-filter

post-filter

[openfeedback](#)



```
PUT starwars
```

```
{  
  "mappings": {  
    "properties": {  
      "text.tokens": {  
        "type": "sparse_vector"  
      }  
    }  
  }  
}  
  
"These are not the droids you are looking for.",  
"Obi-Wan never told you what happened to your father."
```

```
GET starwars/_search
```

```
{  
  "query": {  
    "sparse_vector": {  
      "field": "text.tokens",  
      "query_vector": { "lucas": 0.50047517,  
                        "ship": 0.29860738,  
                        "dragon": 0.5300422,  
                        "quest": 0.5974301, ... }  
    }  
  }  
}
```

[openfeedback](#)





# ELSER

Elastic Learned Sparse Encoder

*sparse\_vector*

Not BM25 or (dense) vector

Sparse vector like BM25

Stored as inverted index

Commercial

## Machine Learning Inference Pipelines

Inference pipelines will be run as processors from the Enterprise Search Ingest Pipeline

**New** Improve your results with ELSER ×

ELSER (Elastic Learned Sparse Encoder) is our **new trained machine learning model** designed to efficiently use context in natural language queries. This model delivers better results than BM25 without further training on your data.

 Deploy

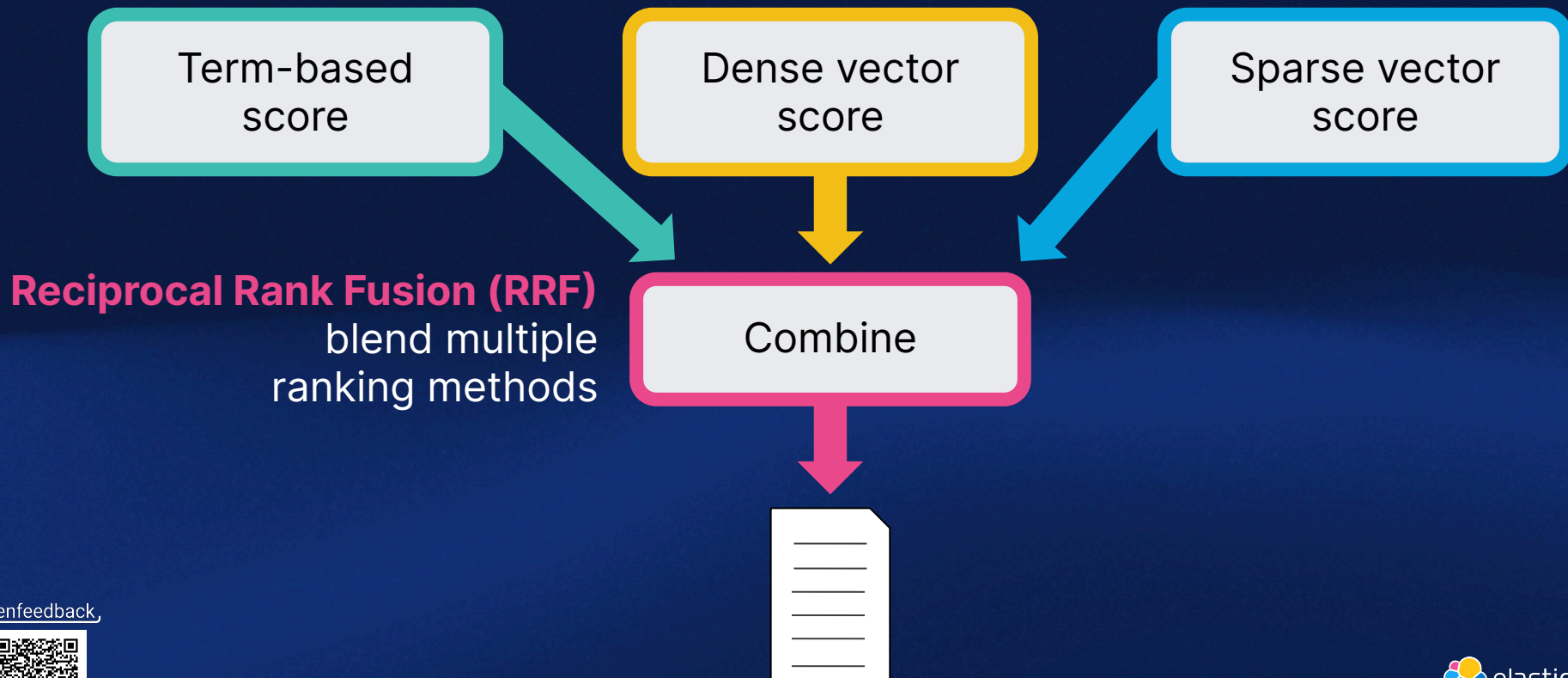
[Learn more](#) 

 Add Inference Pipeline

[Learn more about deploying Machine Learning models in Elastic](#) 



# Hybrid ranking



[openfeedback](#)



# Reciprocal Rank Fusion (RRF)

$$RRFscore(d \in D) = \sum_{r \in R} \frac{1}{k+r(d)}$$

D - set of docs

R - set of rankings as permutation on 1..|D|

k - typically set to 60 by default

Dense Vector			
Doc	Score	r(d)	k+r(d)
A	1	1	61
B	0.7	2	62
C	0.5	3	63
D	0.2	4	64
E	0.01	5	65

BM25			
Doc	Score	r(d)	k+r(d)
C	1,341	1	61
A	739	2	62
F	732	3	63
G	192	4	64
H	183	5	65



Doc	RRF Score
A	$1/61 + 1/62 = 0,0325$
C	$1/63 + 1/61 = 0,0323$
B	$1/62 = 0,0161$
F	$1/63 = 0,0159$
D	$1/64 = 0,0156$

openfeedback



```
GET index/_search
{
  "retriever": {
    "rrf": {
      "retrievers": [{
        "standard" { "query": {
          "match": {...}
        }
      }
    }, {
      "standard" { "query": {
        "sparse_vector": {...}
      }
    }, {
      "knn": { ... }
    }
  ]
}
```

Hybrid Ranking



BM25f

+

Sparse Vector

+

Dense Vector

[openfeedback](#)

# ChatGPT

## Elastic and LLMs



openfeedback



Gen AI

Search engines

openfeedback



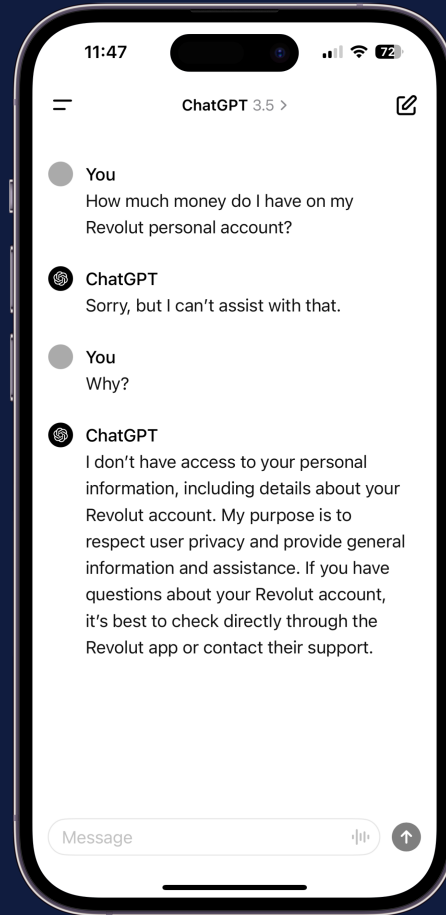
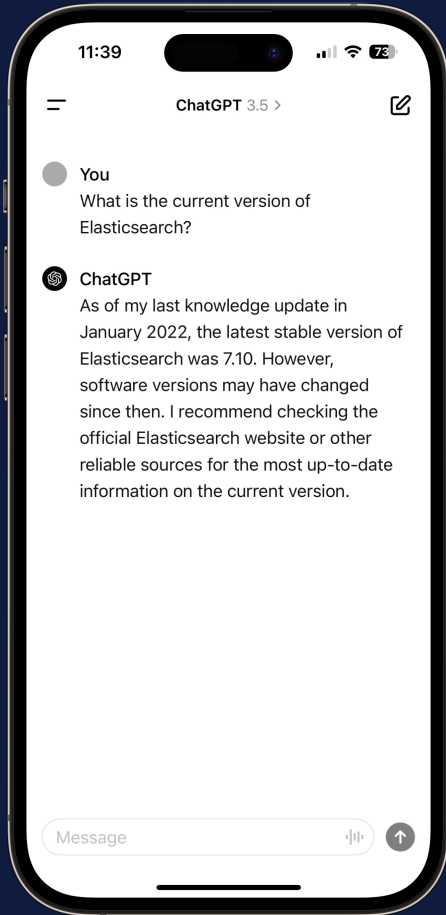


# LLM: opportunities and limits



[openfeedback](#)

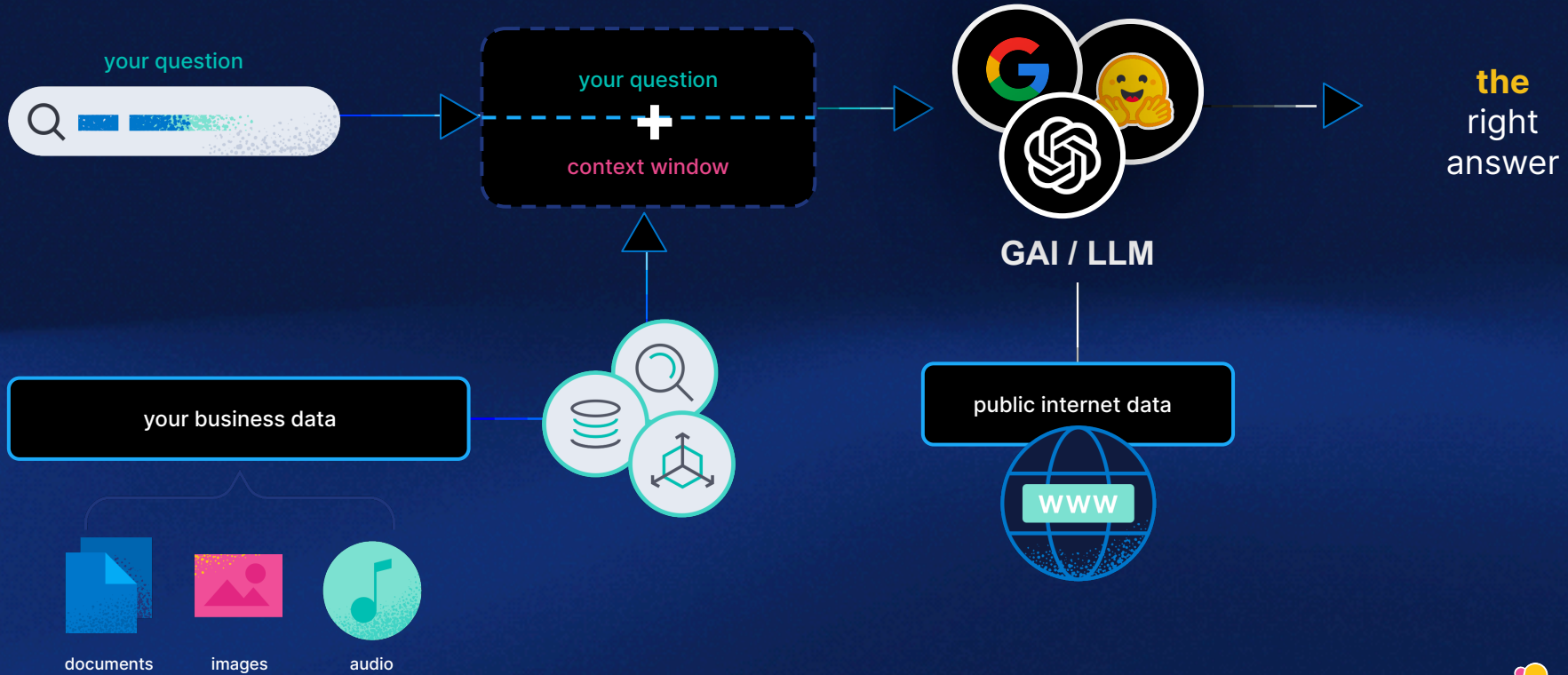




[openfeedback](#)



# Retrieval Augmented Generation





# Demo

## Elastic Playground

[openfeedback](#)



Search your transactions:

This search is not enabled by Elastic and reflects the kind of functionality available to customers today.

[Submit](#)

Date	Account	Description	Value	Opening balance	Closing balance
18/06/24	EL03-130981-Transmission	Inbound payment made from EL03-130981-Transmission, St.james's Plac (STJ): 864dce1b-bb95-47d5-87dd-7d02f3b10c3f	7419.0	-825.0	6594.0
18/06/24	EL03-130981-Transmission	Purchase at merchant: Southeastern Grocers, LLC, location: Fayetteville,AR	82.0	6594.0	6512.0
18/06/24	EL03-130981-Transmission	Purchase at merchant: Müller Holding Ltd. & Co. KG, location: Glendale,AZ	188.0	6512.0	6324.0
17/06/24	EL03-130981-Transmission	Payment made from EL03-130981-Transmission to Elwood Erickson, Mitie Grp. (MTO): d37085fc-1382-4593-9cb8-26e5526bd9a0	533.0	20.0	-513.0
17/06/24	EL03-130981-Transmission	Payment made from EL03-130981-Transmission to Classie Johns, Barclays (BARC): 75b603a2-1c1b-45e9-a7ec-4a551bf98a8d	312.0	-513.0	-825.0
16/06/24	EL03-130981-Transmission	Purchase at merchant: E-MART Inc., location: Fayetteville,AR	31.0	51.0	20.0
14/06/24	EL03-130981-Transmission	Purchase at merchant: Dick's Sporting Goods, Inc., location: Montgomery,AL	182.0	329.0	147.0
14/06/24	EL03-130981-Transmission	Purchase at merchant: Valor Holdings Co., Ltd., location: Louisville,KY	96.0	147.0	51.0
13/06/24	EL03-130981-Transmission	Purchase at merchant: The Save Mart Companies, location:	34.0	363.0	329.0

[openfeedback](#)

# Elasticsearch

You Know, for **Semantic** Search

[openfeedback](#)



# La **météorite GPT** annonce t'elle la disparition des **dinosaures "moteur de recherche"** ?

David Pilato | @dadoonet

openfeedback,



THE DARK  
CHAPTER OF THE  
DEVFEST NANTES  
2024