



Searching for similar musics

David Pilato | [@dadoonet](#)

kpler



Elasticsearch

You Know, for Search



Elasticsearch

Lucene



66

These are not the droids
you are looking for.

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

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

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



These are not the droids you are looking for.

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

These are not the droids you are looking for.



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


"filter": "lowercase"

These

are

not

the

droids

you

are

looking

for



these

are

not

the

droids

you

are

looking

for

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

These

are

not

the

droids

you

are

looking

for



droids

you

looking

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

```
droids  
you  
looking
```



```
droid  
you  
look
```

These are `not` 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
}]}
```



Semantic
search
≠
Literal
matches



similarweb

**YOU'RE COMPARING
APPLES TO NECTARINES**



Elasticsearch

You Know, for Search



Elasticsearch

You Know, for **Vector** Search

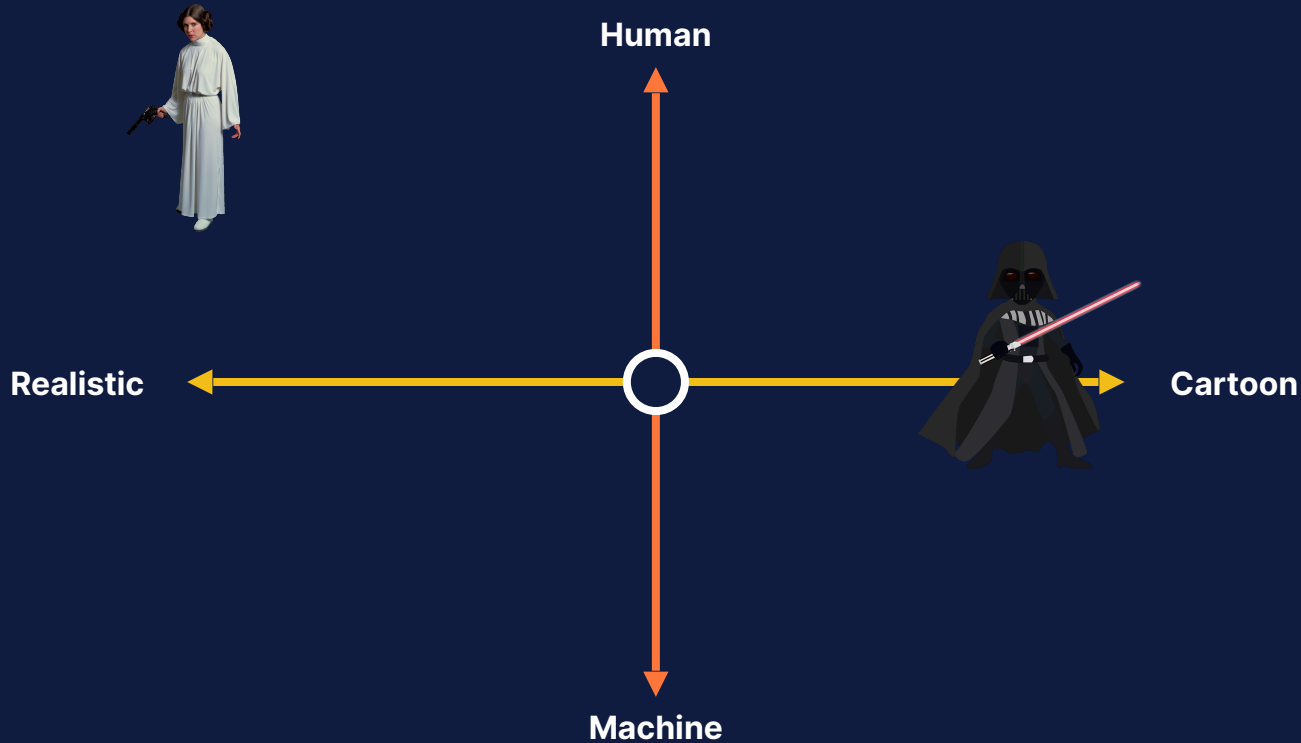
Embeddings represent your data



Example: 1-dimensional vector



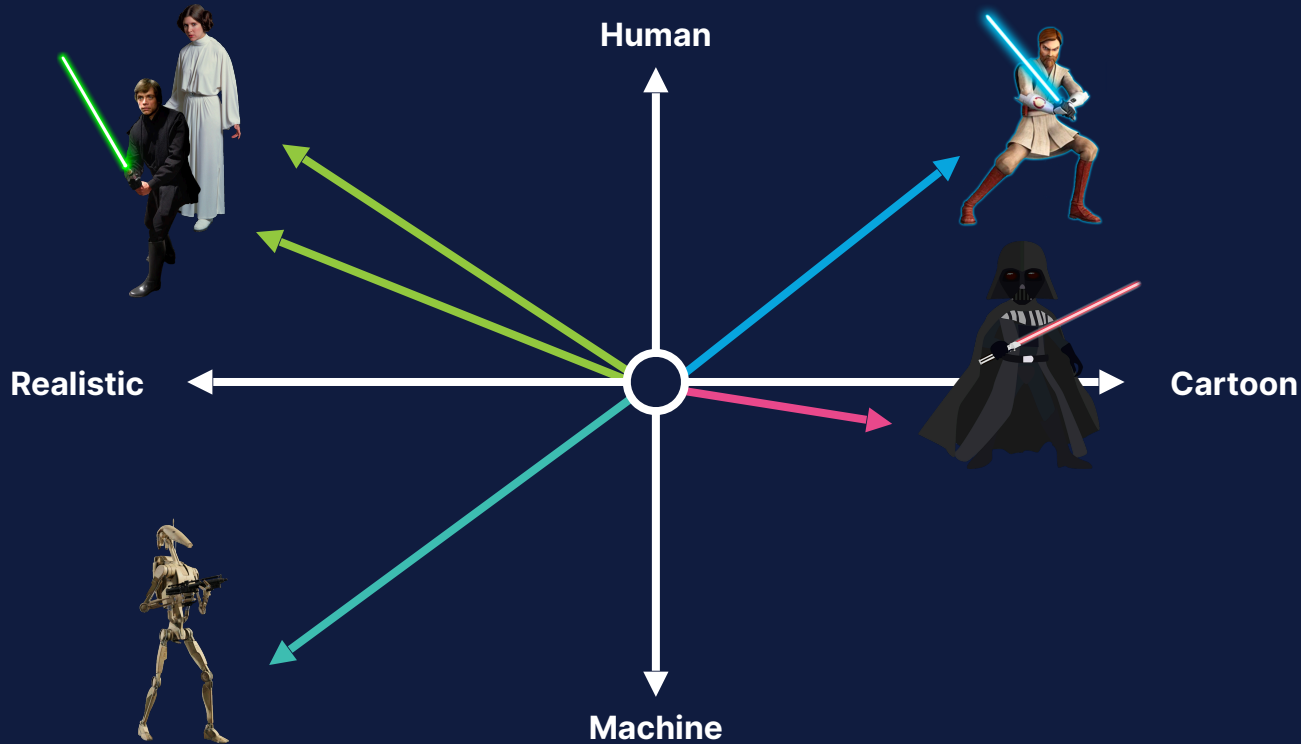
| Character | Vector |
|---|--------|
|  | $[-1]$ |
|  | $[1]$ |






Multiple dimensions represent different data aspects



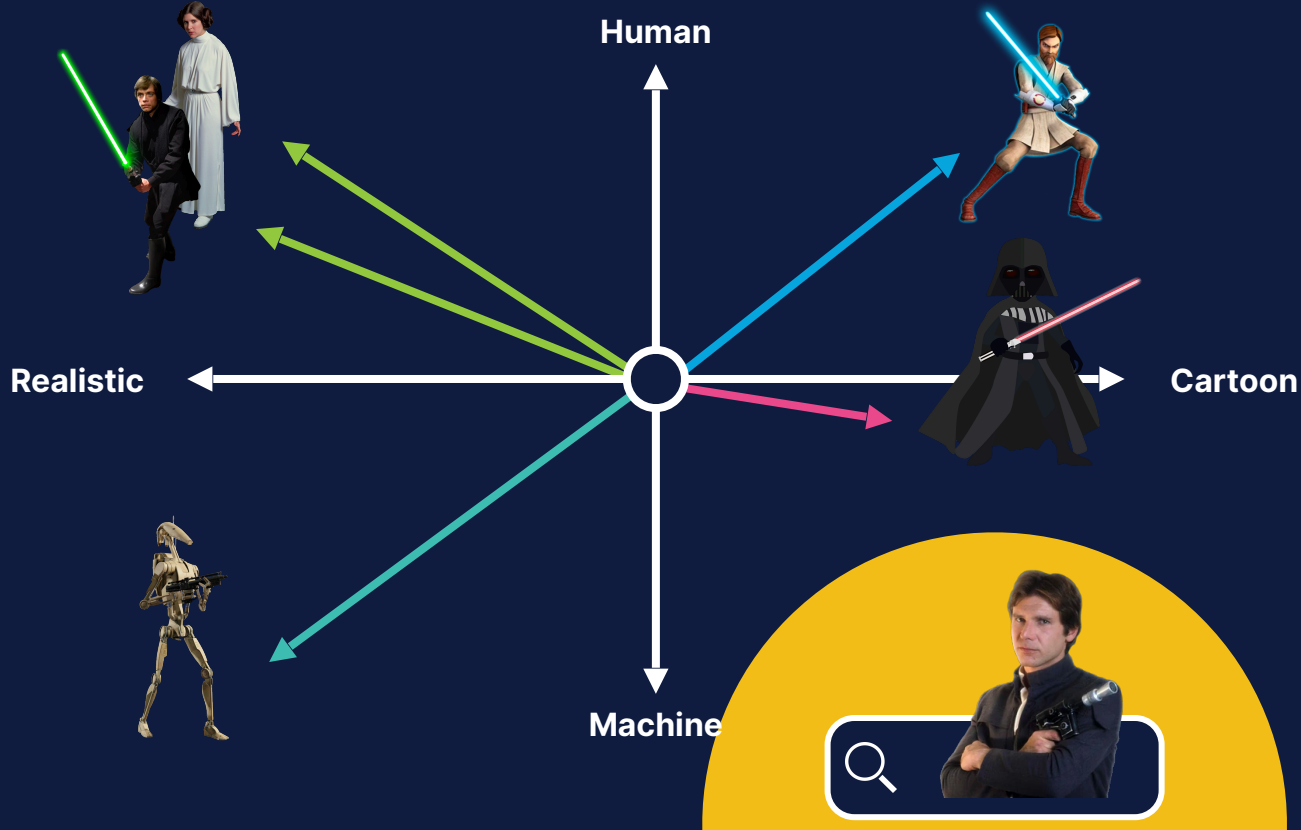
| Character | Vector |
|---|-----------|
|  | $[-1, 1]$ |
|  | $[1, 0]$ |

Similar data is grouped together



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

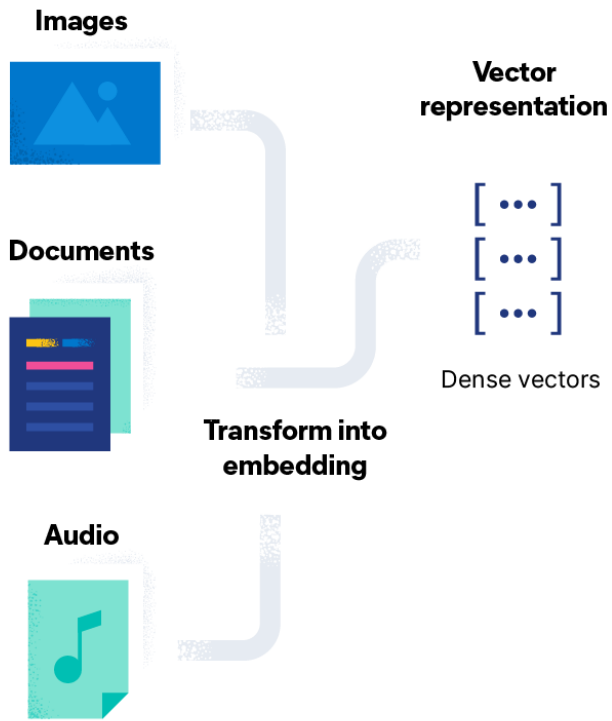
Vector search ranks objects by similarity (~relevance) to the query



| Rank | Result |
|-------|---|
| Query |  |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |

How do you index **vectors**?

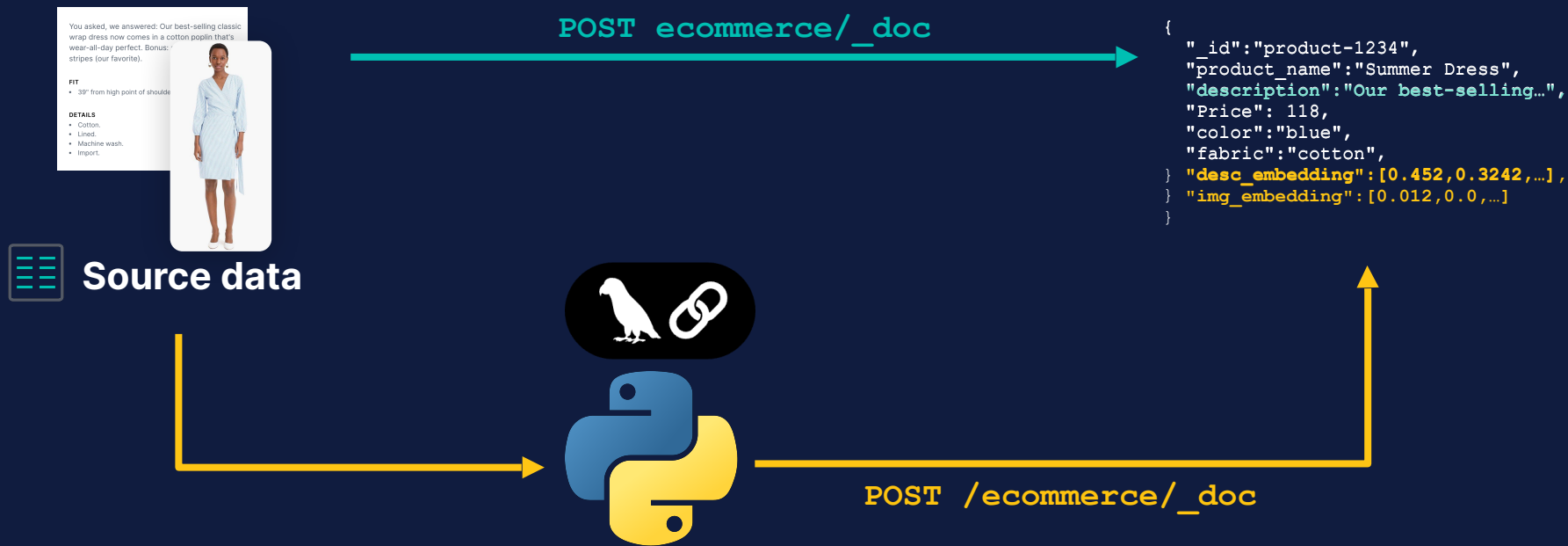
Architecture of Vector Search



dense_vector field type

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

Data Ingestion and Embedding Generation

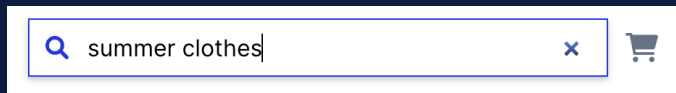


How do you search **vectors**?

Architecture of Vector Search

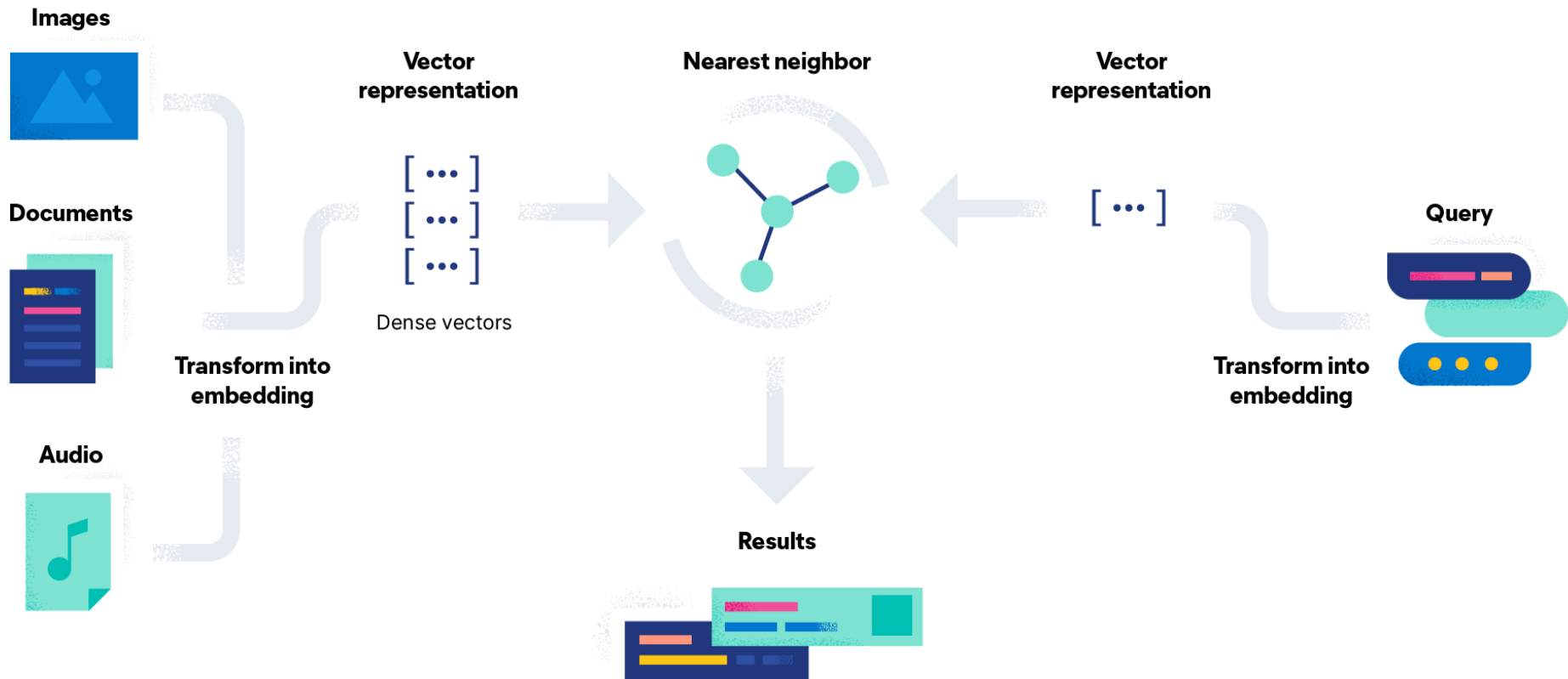


knn query



```
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
}
```

Architecture of Vector Search

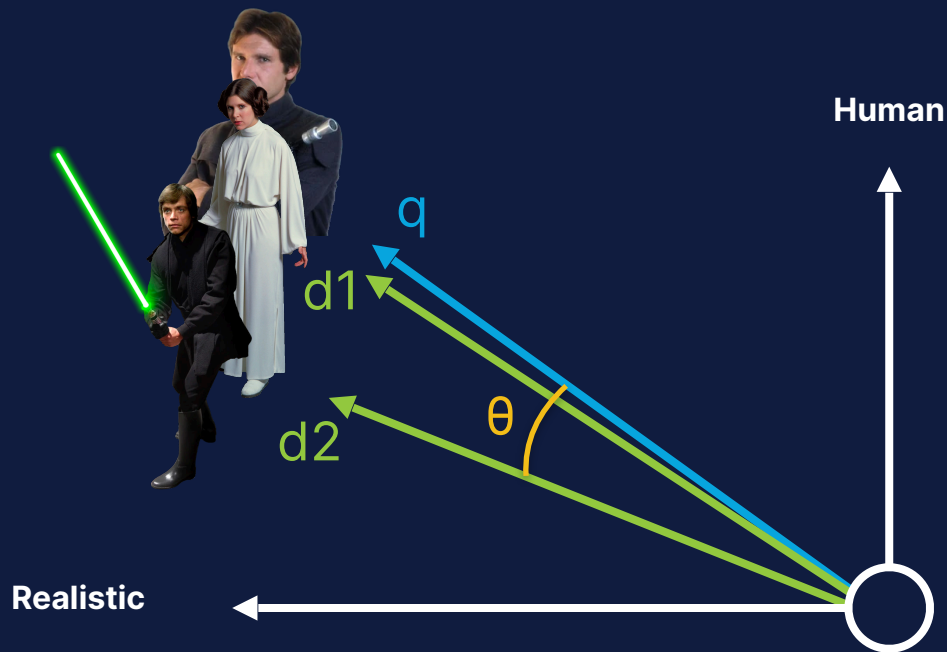




But how does it
really work?



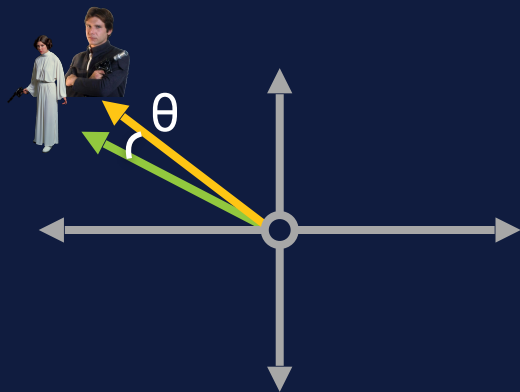
Similarity



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

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

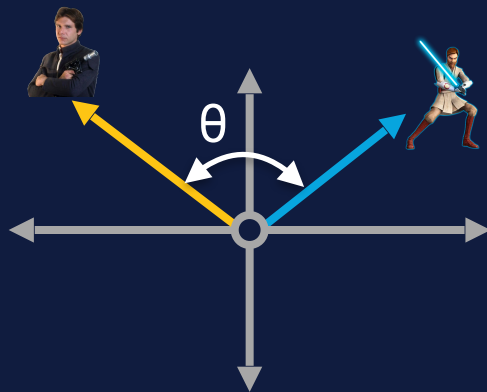
Similarity: cosine (cosine)



Similar vectors

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

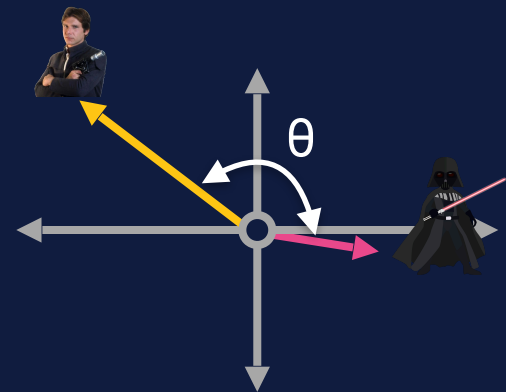
$$\text{_score} = \frac{1 + 1}{2} = 1$$



Orthogonal vectors

θ close to 90°
 $\cos(\theta)$ close to **0**

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



Opposite vectors

θ close to 180°
 $\cos(\theta)$ close to **-1**

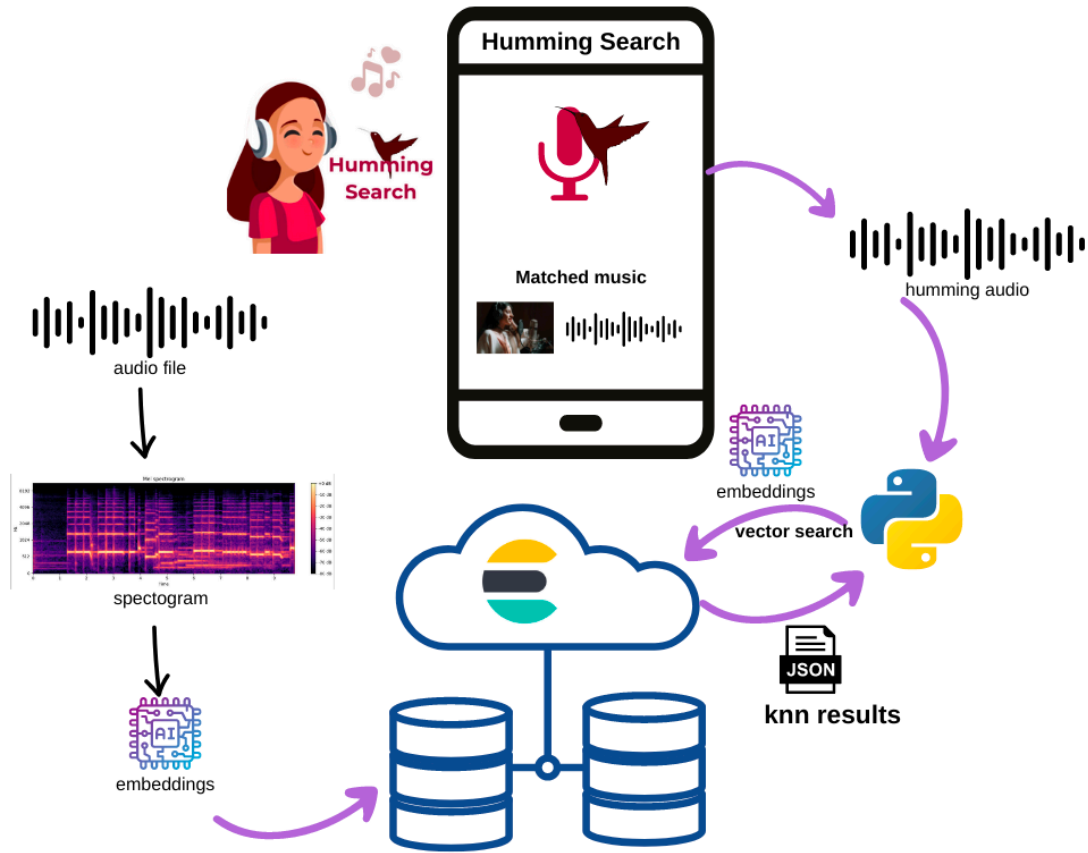
$$\text{_score} = \frac{1 - 1}{2} = 0$$



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



16/09/2023



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



Searching for similar musics

David Pilato | [@dadoonet](#)

kpler

