Agenda

- "Classic" search and its limitations
- ML model and usage
- Vector search or hybrid search in Elasticsearch
- OpenAI's ChatGPT or LLMs with Elasticsearch

bigdata

elastic

Elasticsearch

You Know, for Search

"

These are not the droids you are looking for.

bigdata

elastic

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

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

```json
{ "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
}]}
```

bigdata

elastic

TODAY

🔍 *X-wing starfighter squadron*

TOMORROW

🔍 *What ships and crews do I need to destroy an almost finished death star?*
Or is there a secret weakness?

bigdata

elastic

Elasticsearch

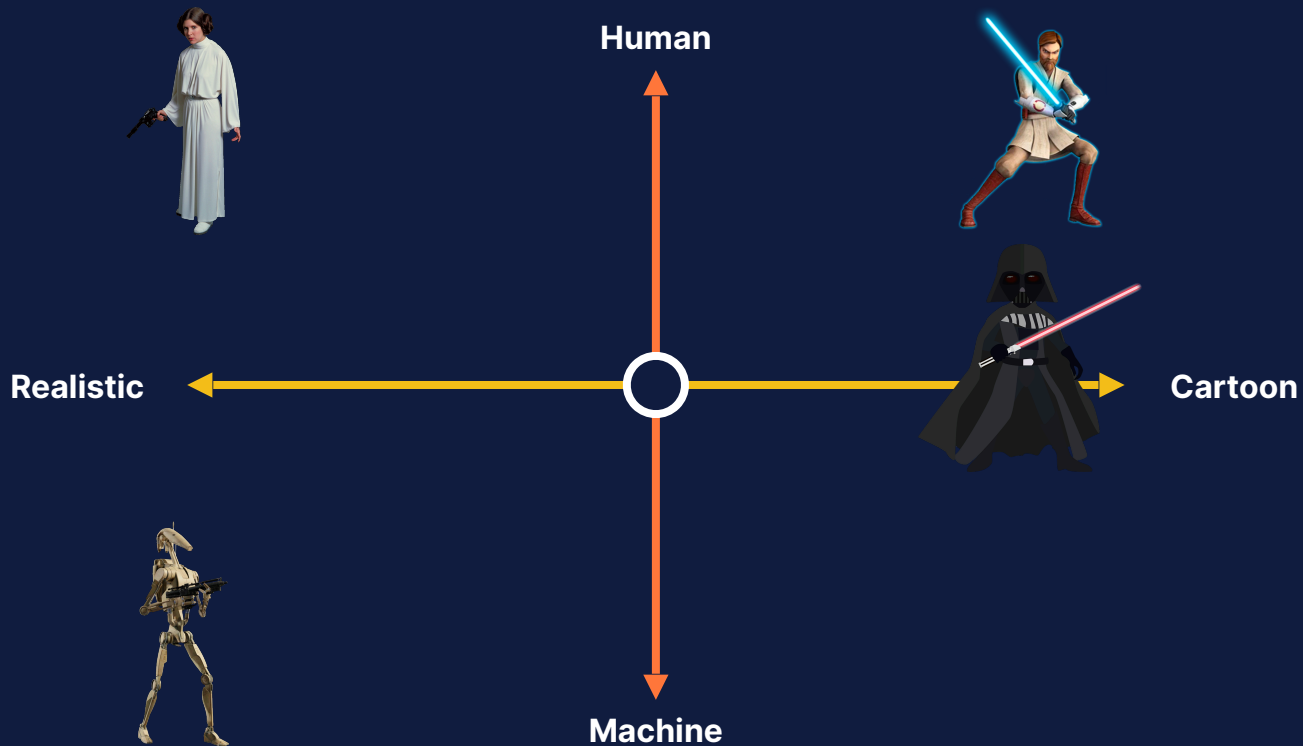You Know, for Search

What is a **Vector**?

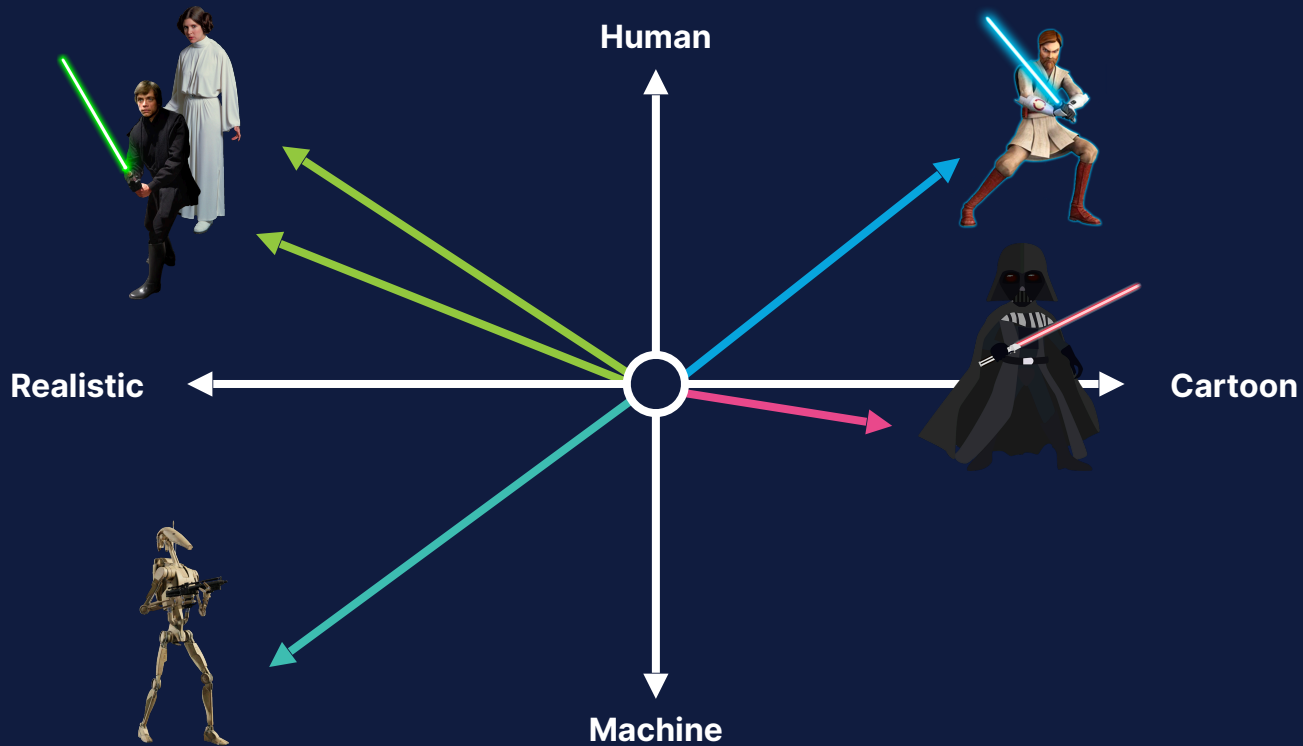# Embeddings represent your data
Example: 1-dimensional vector

Realistic ←————————————○————————————→ **Cartoon**

Character	Vector
	[-1]
	[1]

bigdata

elastic

Multiple dimensions
represent different data aspects

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

Similar data
is grouped together

Human

Realistic

Cartoon

Machine

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

bigdata

elastic

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

Human

Realistic

Cartoon

Machine
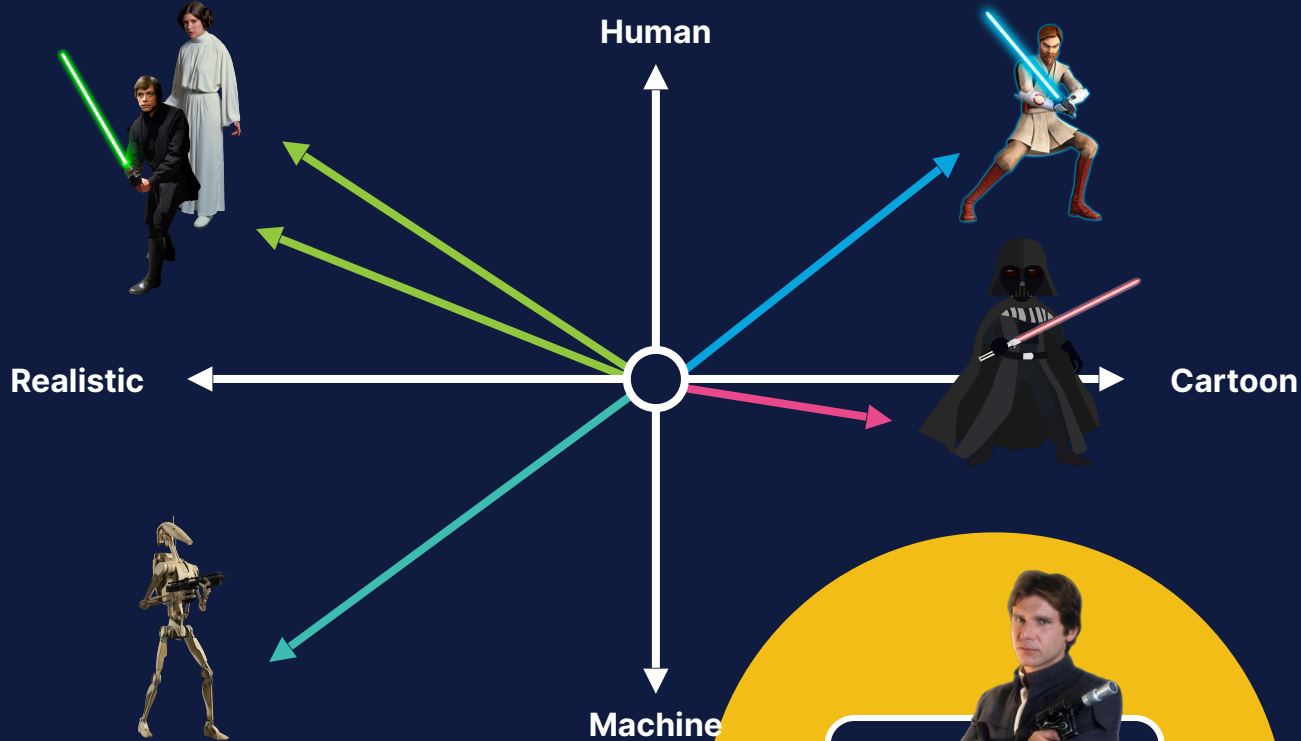
Rank	Result
Query	
1	
2	
3	
4	
5	

bigdata

elastic

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

bigdata

elastic

Problem
training vs actual use-case
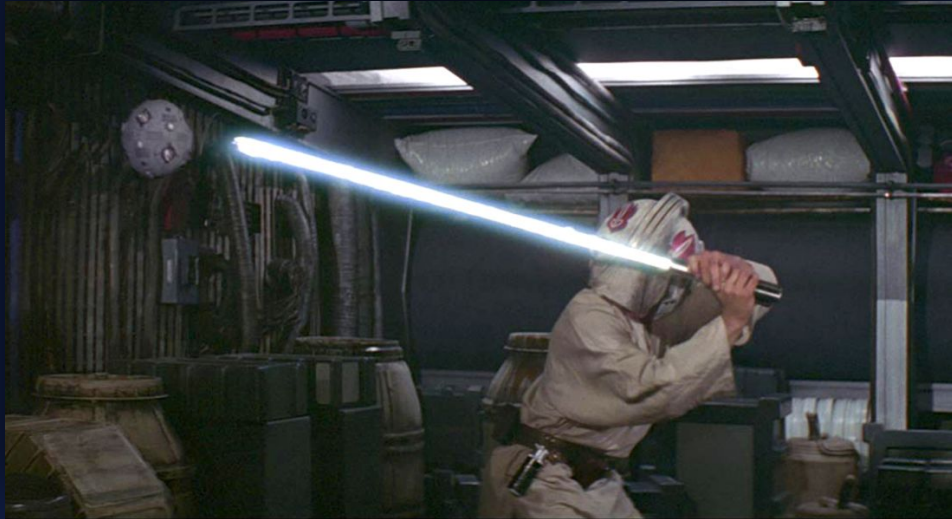
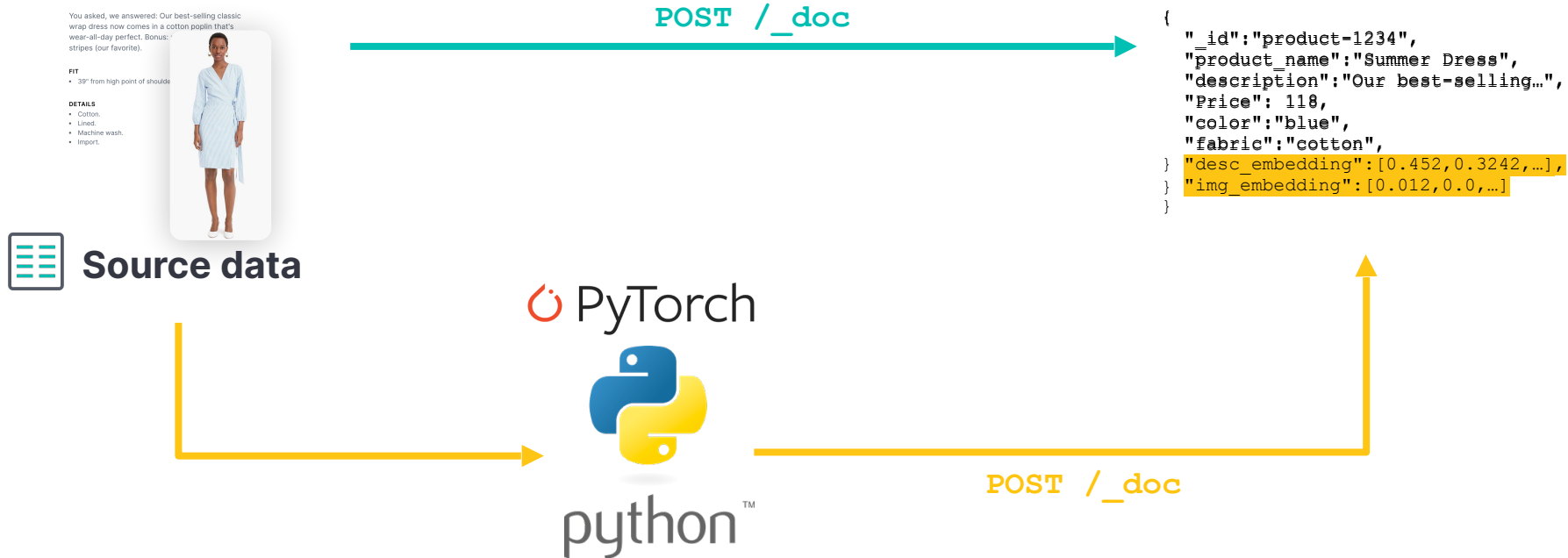# Architecture of Vector Search

Images

Vector representation

Documents

$$[\; \cdots \;]$$

$$[\; \cdots \;]$$

$$[\; \cdots \;]$$

Transform into embedding

Dense vectors

Audio
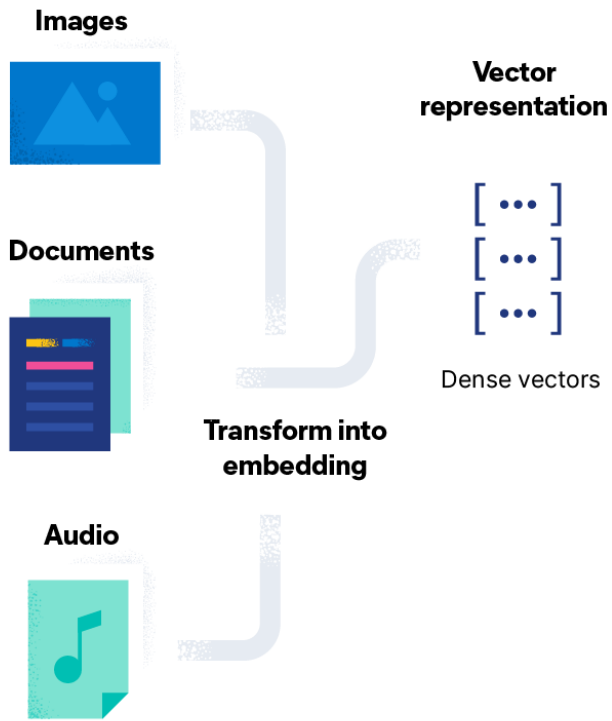
Data Ingestion and Embedding Generation

POST /_doc

You asked, we answered: Our best-selling classic wrap dress now comes in a cotton poplin that's wear-all-day perfect. Bonus: stripes (our favorite).

FIT
• 39" from high point of shoulder

DETAILS
• Cotton.
• Lined.
• Machine wash.
• Import.

Source data

```
{
    "_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,…]
}
```

PyTorch

python™

POST /_doc

bigdata

elastic

With Elastic ML

Source data

POST /_doc

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

ML Inference pipelines

⊕ Add inference pipeline

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

ml-inference-embedding-generation   Actions ⋮
● Deployed   pytorch   text_embedding

ml-inference-emotional-analysis   Actions ⋮
● Deployed   pytorch   text_classification

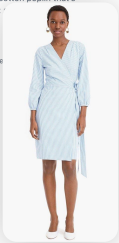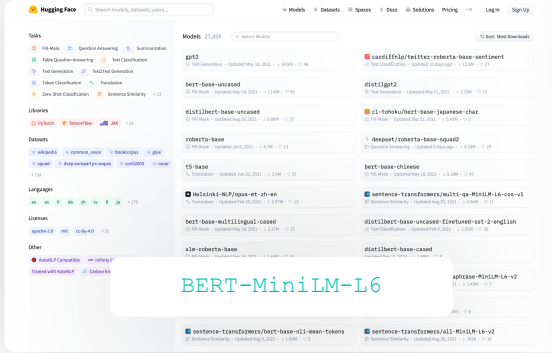Learn more about deploying ML models in Elastic ⧉

You asked, we answered: Our best-selling classic wrap dress now comes in a cotton poplin that's wear-all-day perfect. Bonus: stripes (our favorite).

FIT
• 39" from high point of shoulder

DETAILS
• Cotton.
• Lined.
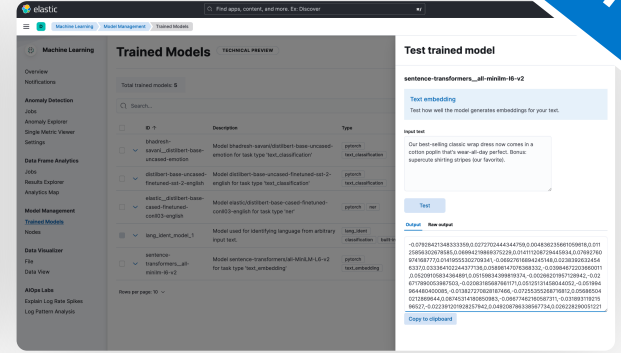• Machine wash.
• Import.

bigdata

elastic

Eland Imports PyTorch Models

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

Select the
appropriate model

Load it

Manage models

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
- MF
- Ro

Third party text classification models

- BE
- De
- Dis
- Dis
- Fin
- **Dis**
- Tw

Third party named entity recognition models

- BER
- **Dis**
- Dis
- Dis
- **Dis**
- Dis

Third party question answering models

- **All**
- BER
- Dist
- Elec
- **Mul**
- Tiny

Third party text embedding models

Text
betw
euc

The
type

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

bigdata

Full list at: ela.st/nlp-supported-models

elastic

How do you

search **vectors**?

Vector Query

summer clothes

PyTorch

python
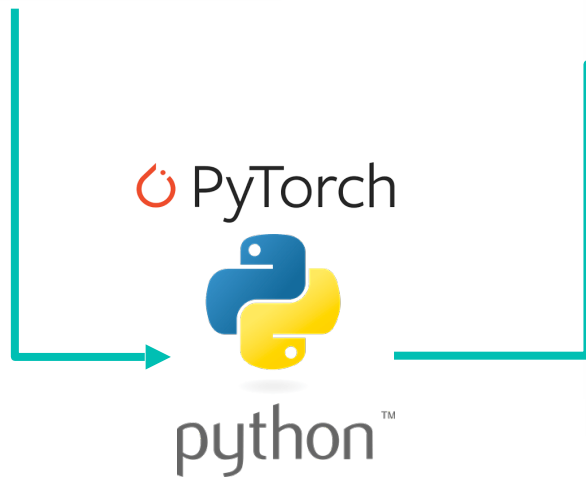
```
GET product-catalog/_search
{
  "knn": {
    "field": "desc_embbeding",
    "k": 5,
    "num_candidates": 50,
    "query_vector": [0.123, 0.244,...],


    "filter": {
      "term": {
        "department": "women"
      }
    }
  },
  "size": 10
}
```

bigdata

elastic

Vector Query

summer clothes

```
GET product-catalog/_search
{
  "knn": {
    "field": "desc_embbeding",
    "k": 5,
    "num_candidates": 50,
    "query_vector_builder": {
      "text_embedding": {
        "model_text": "summer clothes",
        "model_id": <text-embedding-model>
      },

    "filter": {
      "term": {
        "department": "women"
      }
    }
  },
  "size": 10
}
```

Transformer model

PyTorch

bigdata

elastic

Vector Search
components

Search	Index	Generate
Query	**Mapping**	**Embedding**
kNN	dense_vector	Text embedding model (3rd party, local, in Elasticsearch)

bigdata

elastic
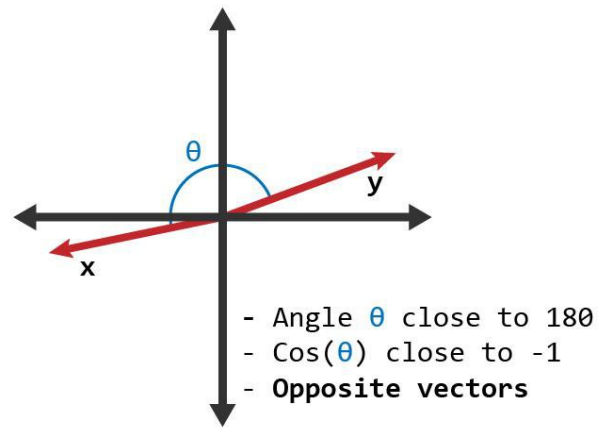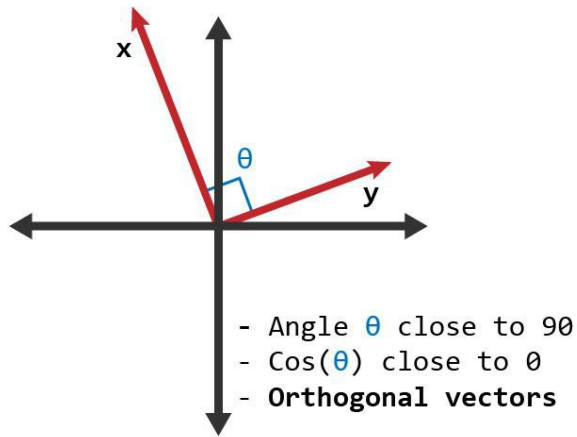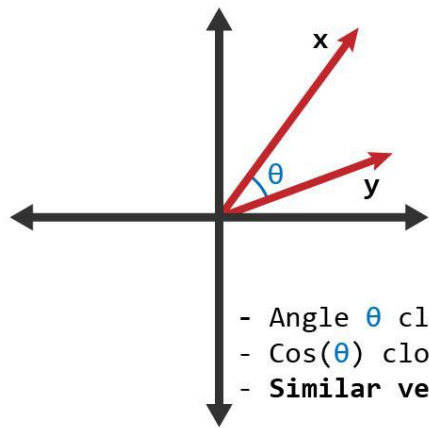
But how does it

really work?

Similarity: cosine (cosine)

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

$$\_score = \frac{1 + cos(\theta)}{2}$$
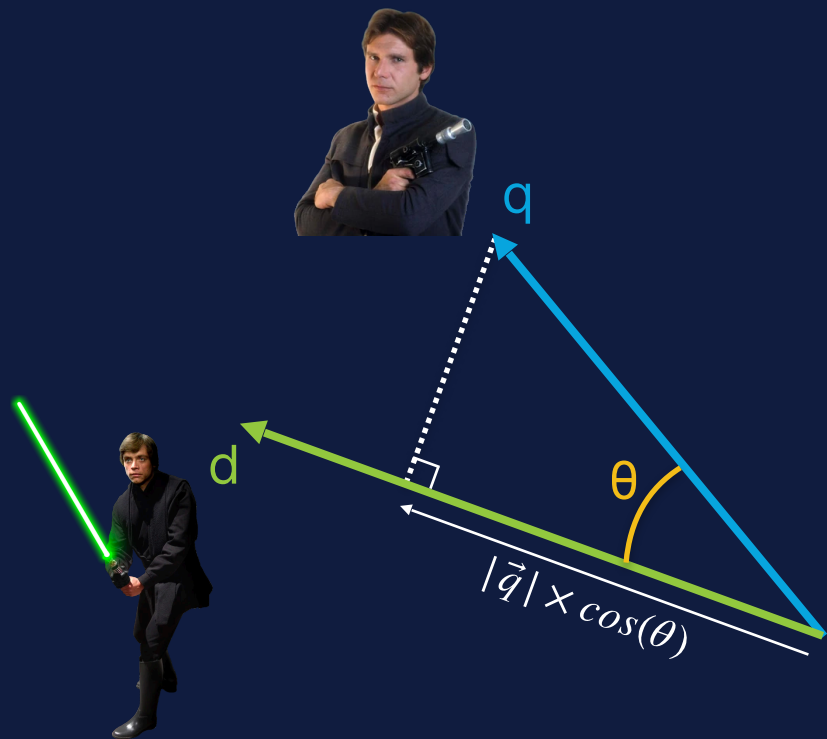
Similarity: cosine (cosine)

- Angle θ close to 0
- Cos(θ) close to 1
- **Similar vectors**

- Angle θ close to 90
- Cos(θ) close to 0
- **Orthogonal vectors**

- Angle θ close to 180
- Cos(θ) close to -1
- **Opposite vectors**
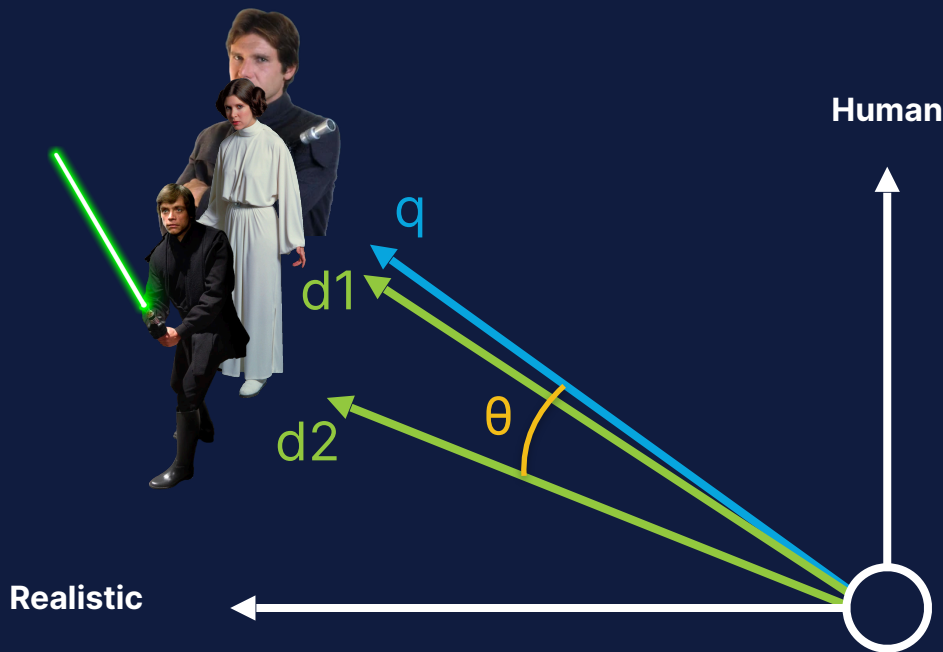
$$\_score = \frac{1+1}{2} = 1$$

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

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

bigdata

elastic
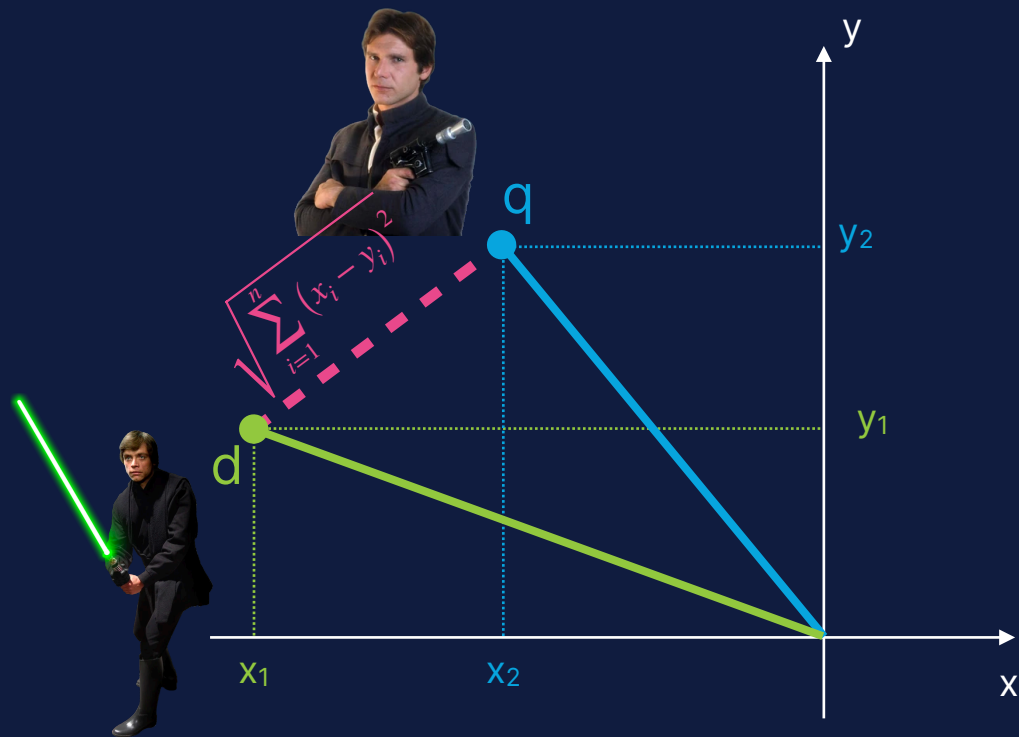
Similarity: Dot Product (dot_product)

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

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

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

Similarity: Euclidean distance (`l2_norm`)

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

$$\_score = \frac{1}{1 + (l2\_norm_{q,d})^2}$$

bigdata

elastic

Brute Force

bigdata

elastic

Hierarchical Navigable Small Worlds (HNSW)
One popular approach

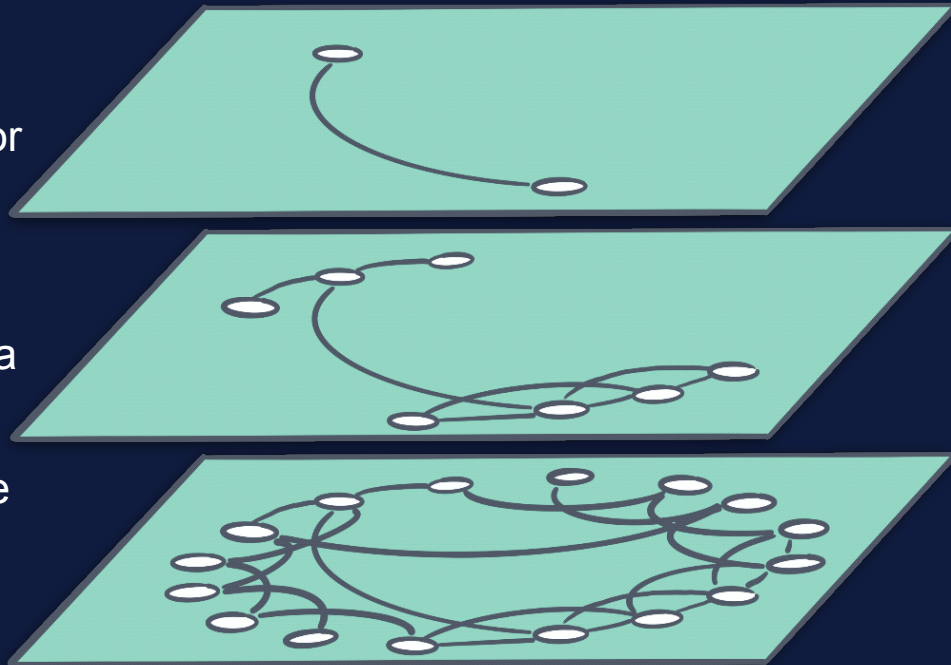✅ **HNSW:** a layered approach that simplifies access to the nearest neighbor

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

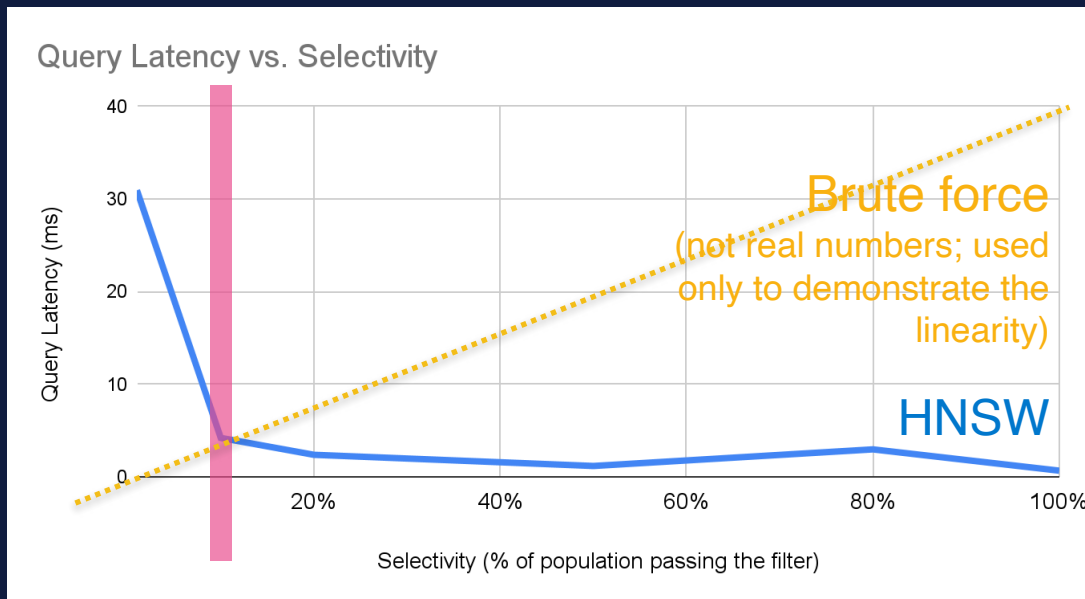✅ **Balance:** Bartering a little accuracy for a lot of scalability

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

bigdata

elastic

Filtering KNN Vector Similarity
Automatically choose between brute force and HNSW

Query Latency vs. Selectivity

Brute force
(not real numbers; used only to demonstrate the linearity)

HNSW

Query Latency (ms): 40, 30, 20, 10, 0

Selectivity (% of population passing the filter): 20%, 40%, 60%, 80%, 100%

Bound worst case to 2*(brute force)
- Brute force scales O(n) of filtered
- HNSW scales ~O(log(n)) of all docs

bigdata

elastic

Elasticsearch + Lucene = fast progress ❤️

Increase max number of vector dims to 2048 #95257

Increase the max vector dims to 4096 #99682

Merged mayya-sharipova merged 2 commits into `elastic:main` from `mayya-sharipova:increase_vector_dims_4096` 📋 on

💬 Conversation 5   ⚬ Commits 2   ☑ Checks 0   ± Files changed 8

mayya-sharipova commented on Sep 19   Contributor ···

No description provided.

Increase the max vector dims to 4096   ✕ 3f97c5f

mayya-sharipova added `>enhancement` `:Search/Vectors` `v8.11.0` labels on Sep 19

bigdata
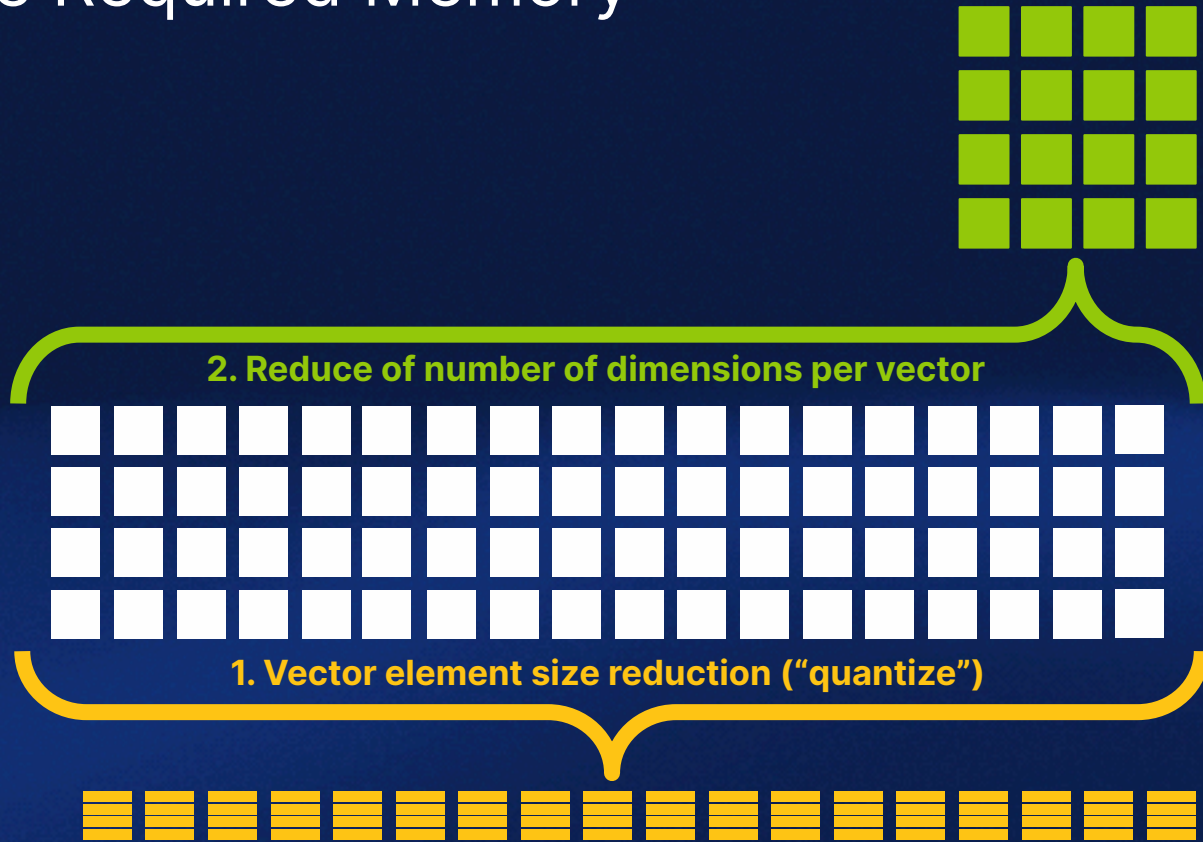
elastic

Scaling Vector Search

Vector search

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

* Continuous improvements in Lucene + Elasticsearch

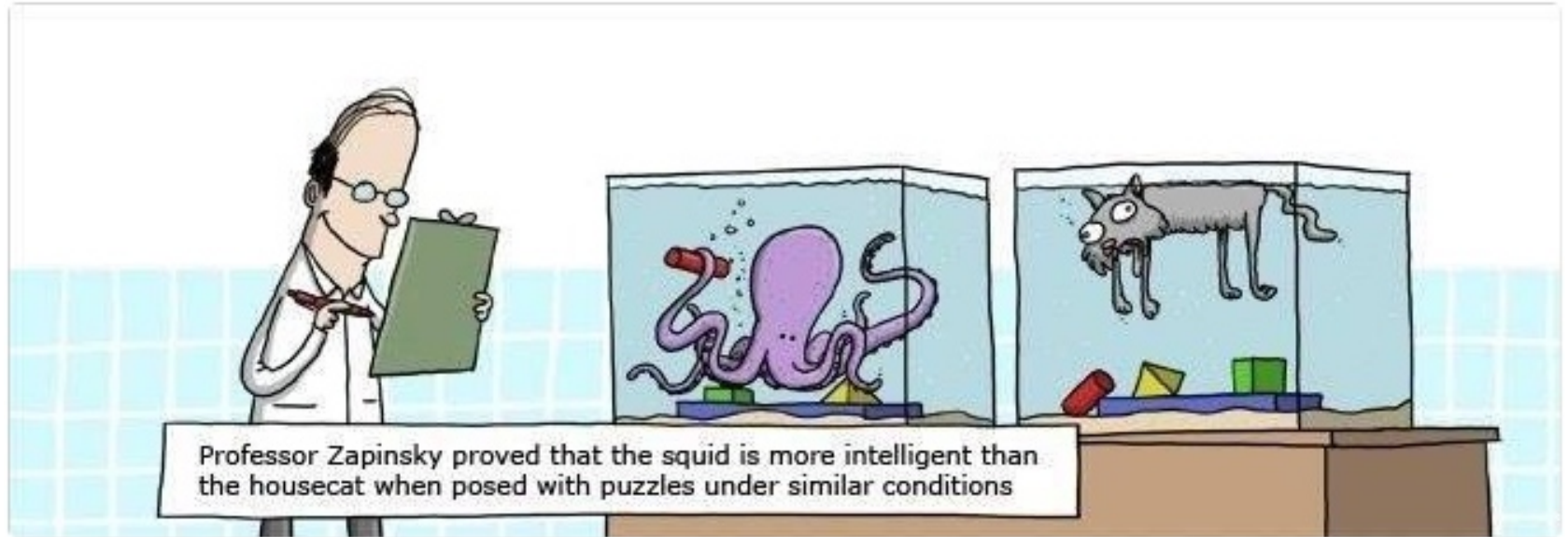## Best practices

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

bigdata

Reduce Required Memory

2. Reduce of number of dimensions per vector

1. Vector element size reduction ("quantize")

bigdata

elastic

Benchmarking

Professor Zapinsky proved that the squid is more intelligent than the housecat when posed with puzzles under similar conditions

https://github.com/erikbern/ann-benchmarks

Add Elasticsearch KNN #401

Merged **erikbern** merged 1 commit into `erikbern:main` from `ceh-forks:elasticsearch` last week

💬 Conversation 5   Commits 1   Checks 34   Files changed 4

ceh commented last week · edited ▾   Contributor ⋯

Add an implementation for Elasticsearch KNN search. Fixes #298.

- Supports specifying `index_options` via arg groups, with one entry for Elasticsearch's default settings for M and EF.
- Supports specifying `num_candidates` via query args.
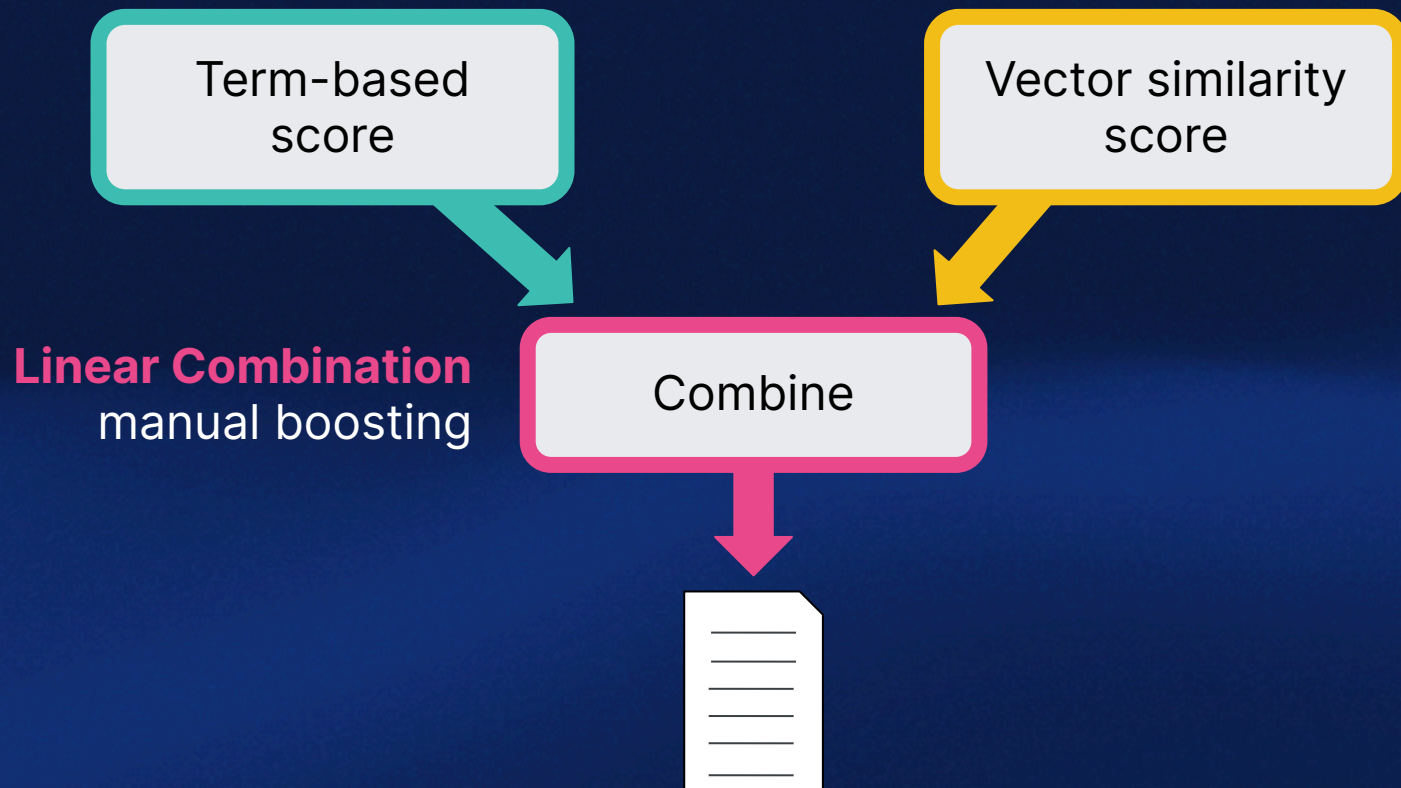
What do you think **@erikbern**, **@alexklibisz**?

😊   👍 1

☁️ bigdata   🌸 elastic

Elasticsearch

You Know, for **Hybrid** Search

bigdata

elastic

```
GET product-catalog/_search
{
  "query": {
    "match": {
      "description": {
        "query": "summer clothes",
        "boost": 0.9
      }
    }
  },
  "knn": {
    "field": "desc_embbeding",
    "query_vector": [0.123, 0.244,...],
    "k": 5,
    "num_candidates": 50,
    "boost": 0.1,
    "filter": {
      "term": {
        "department": "women"
      }
    }
  },
  "size": 10
}
```

```
GET product-catalog/_search
{
  "query": {
    "match": {
      "description": {
        "query": "summer clothes",
        "boost": 0.9
      }
    }
  },
  "knn": [ {
    "field": "image-vector",
    "query_vector": [54, 10, -2],
    "k": 5,
    "num_candidates": 50,
    "boost": 0.1
  },
  {
    "field": "title-vector",
    "query_vector": [1, 20, -52, 23, 10],
    "k": 10,
    "num_candidates": 10,
    "boost": 0.5
  }],
  "size": 10 }
```

bigdata

elastic

```
POST /_ingest/pipeline/_simulate
{
    "pipeline":{"processors":[
        {
            "inference":{"model_id":".elser_model_1"}
        }
    ]},
    "docs": [
        {"_source":{"text_field":"These are not the droids you
        are looking for."}},
        {"_source":{"text_field":"Obi-Wan never told you what
        happened to your father."}},
        {"_source":{"text_field": "No. I am your father!"}}
    ]
}
```

" "

These are not the droids you are looking for.

```
"ml": {

        "inference": {

          "predicted_value": {

            "lucas": 0.50047517,

            "ship": 0.29860738,

            "dragon": 0.5300422,

            "quest": 0.5974301,

            "dr": 2.1055143,

            "space": 0.49377063,

            "robot": 0.40398192,

            ...
```

bigdata

elastic

```
GET product-catalog/_search
{
  "sub_searches": [
    {
      "query": {
        "match": {...}                          BM25f
      }
    },
    {                                             +
      "query": {
        "text_expansion": {...}                 ELSER
      }
    }                                             +
  ],
  "knn": {...},                                  Vector
  "rank": {
    "rrf": {
      "window_size": 50,
      "rank_constant": 20                  Hybrid Ranking
    }
  }
}
```

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|
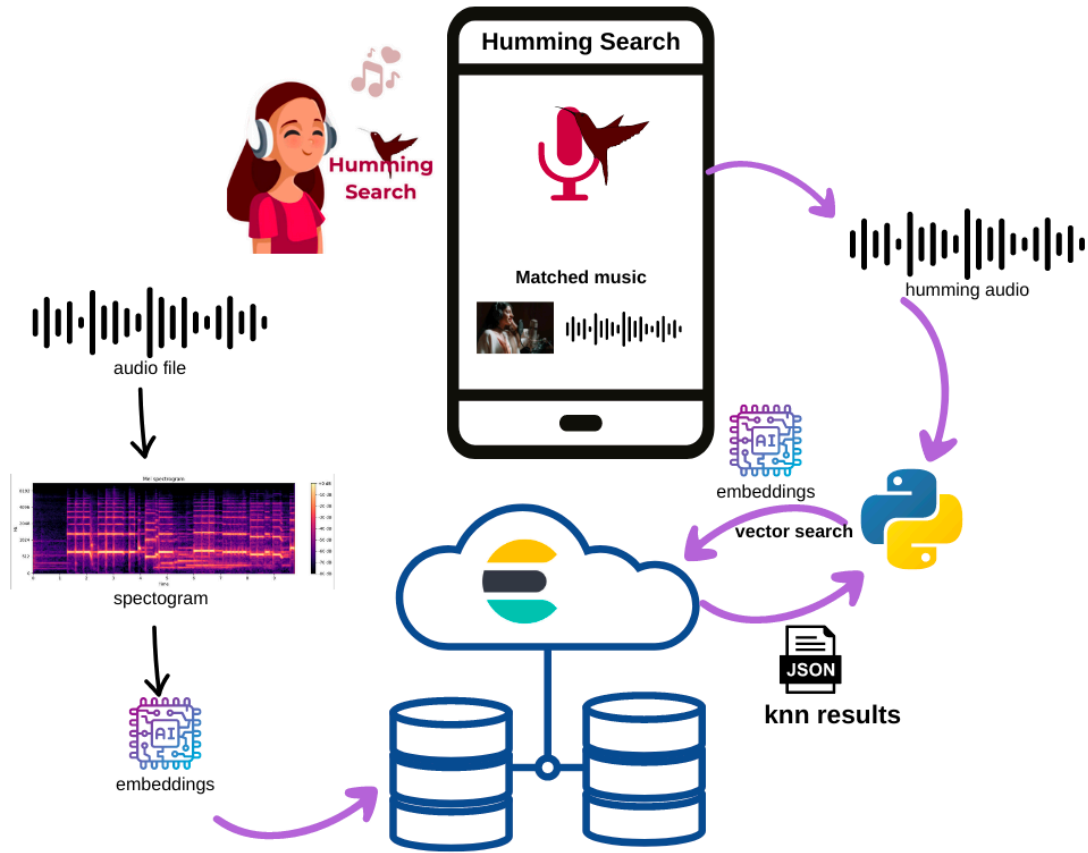k - typically set to 60 by default

Ranking Algorithm 1			
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

Ranking Algorithm 2			
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

bigdata

elastic

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

ChatGPT

Elastic and LLM

LLM: opportunities and limits

your question

your question

GAI / LLM

public internet data

WWW

one
answer

bigdata
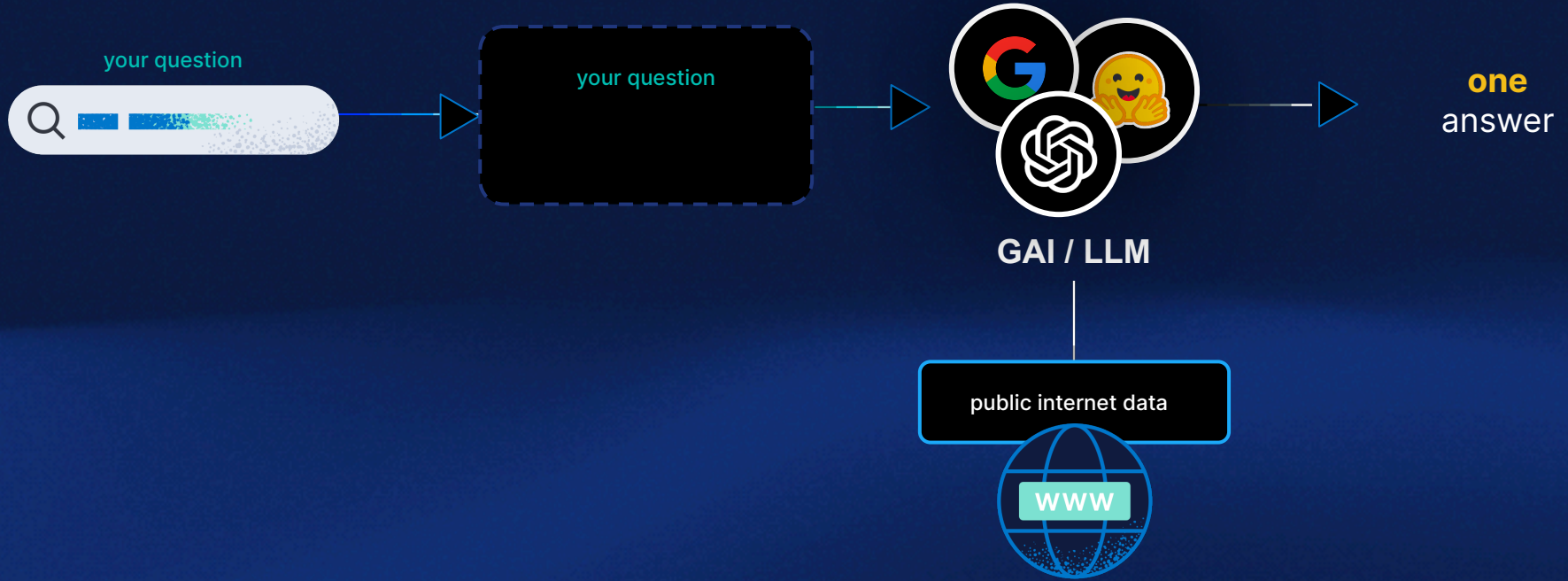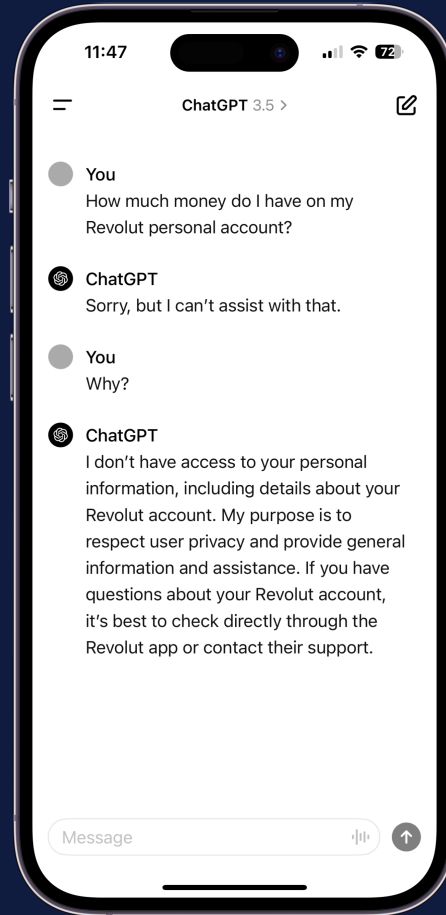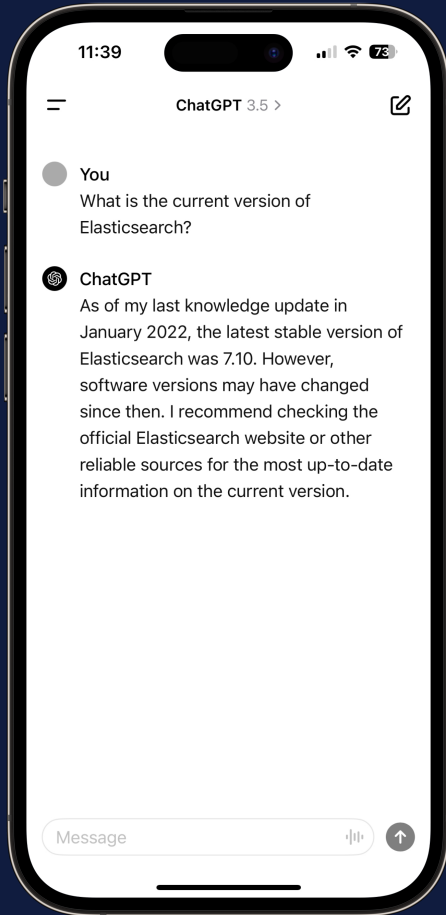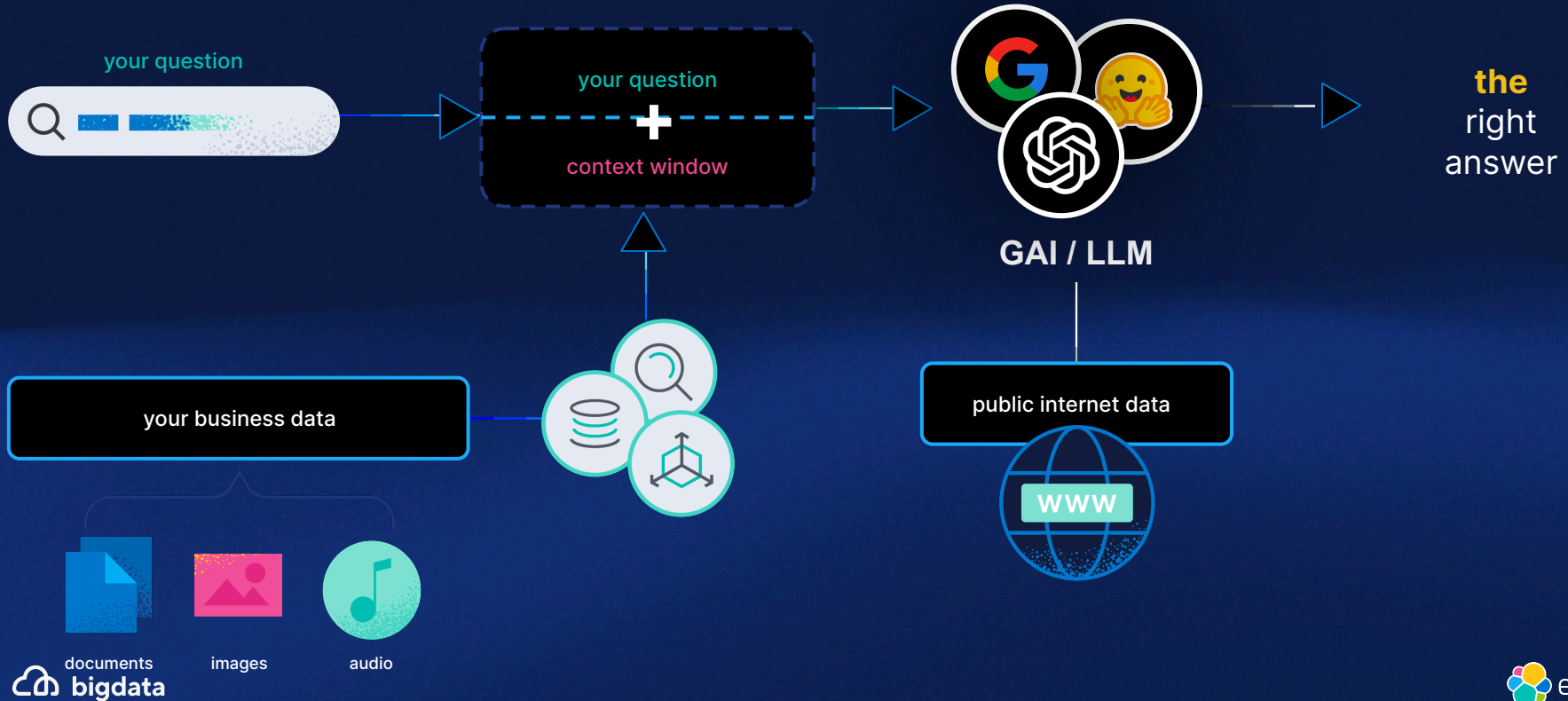
elastic

Demo

Elastic + Azure OpenAI
AWS Bedrock
Google Vertex AI

bigdata

elastic

Conclusion