Go-ing Gopher Hunting with Elasticsearch and Go

Carly Richmond
𝕏 @CarlyLRichmond

Elasticsearch:
You Know, for Search

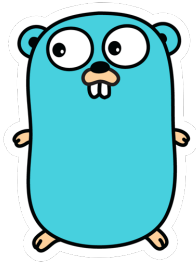

elasticsearch
Lucene

Elasticsearch Go Client

2 Supported API Types:

- Low-level API
- Fully Typed API

Latest version: v8.9
Compatible with Go version 1.13+



SCAN ME

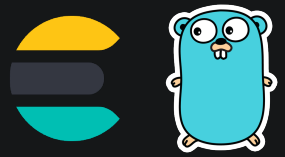
[go-elasticsearch | GitHub]
[go-elasticsearch/v8 | pkg.go.dev]

elastic

Elasticsearch Go Client

search.go

```go
// Elasticsearch init
var cloudID = os.Getenv("ELASTIC_CLOUD_ID")
var apiKey = os.Getenv("ELASTIC_API_KEY")

var client, err =
elasticsearch.NewTypedClient(elasticsearch.Config{
    CloudID: cloudID,
    APIKey:  apiKey,
})
```

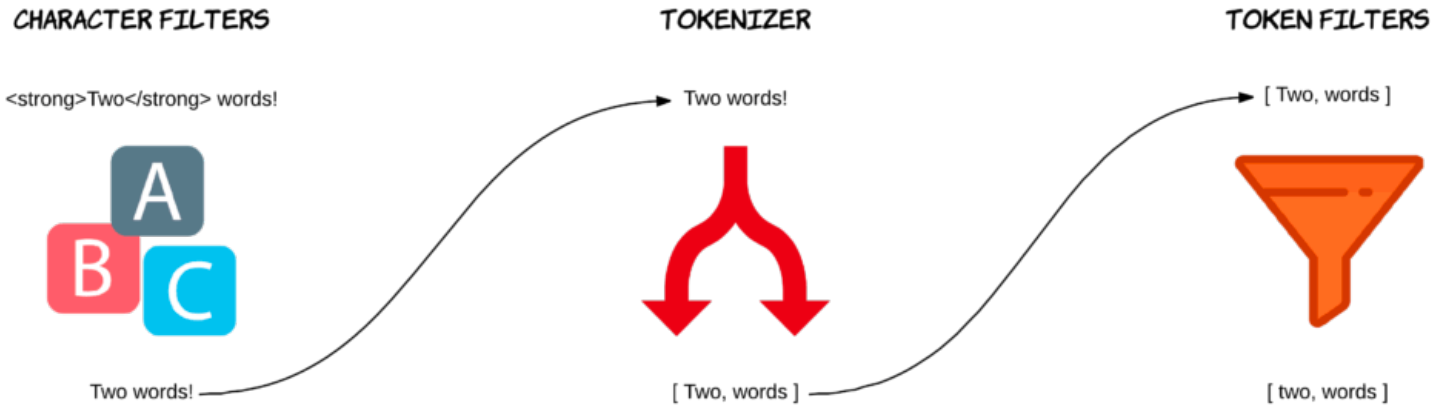


elastic

Elasticsearch:
You Know, for Keyword Search

Credit: Understanding Analysis in Elasticsearch (Analyzers) Published on May 5, 2018 by Bo Andersen

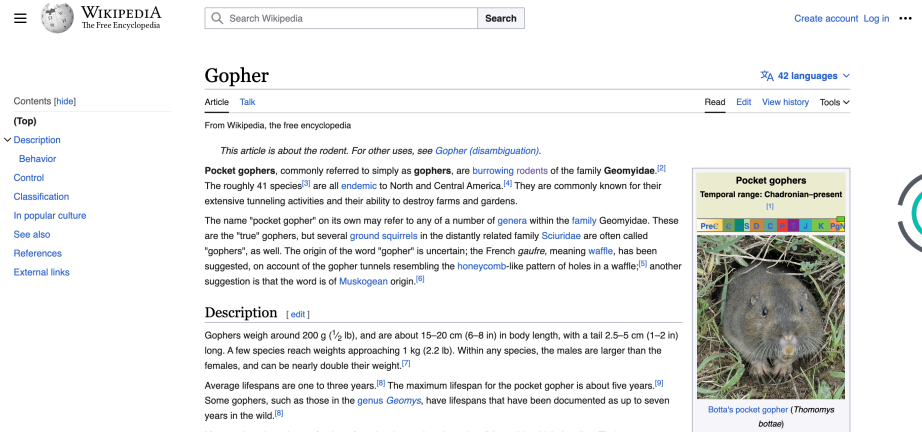
2



Credit: Understanding Analysis in Elasticsearch (Analyzers) Published on May 5, 2018 by Bo Andersen

elastic

Document



gopher.json


```
{
  "_index": "search-rodents",
  "_id": "64f74ecd4acb3df024d91112",
  "_score": 1.7641908,
  "_ignored": ["body_content.enum"],
  "_source": {
    "last_crawled_at": "2023-09-13T18:25:06Z",
    "additional_urls": ["https://en.wikipedia.org/wiki/Gopher"],
    "body_content": "Pocket gophers, commonly referred to simply as gophers, are burrowing rodents of the family Geomyidae.",
    "domains": ["https://en.wikipedia.org"],
    "title": "Gopher - Wikipedia",
    "url": "https://en.wikipedia.org/wiki/Gopher"
  }
}
```


[Source: Elastic Web Crawler]
[Alternative: Exercise: Web Crawler | Tour of Go]

elastic

Keyword Search

keyword-search.go

```go
// Traditional keyword search example
func KeywordSearch(term string) []Rodent {
    res, err := client.Search().
        Index("search-rodents").
        Query(&types.Query{
            Match: map[string]types.MatchQuery{
                "title": {Query: term},
            },
        }).
        From(0).
        Size(10).
        Do(context.Background())

    if err != nil {
        log.Fatal(err)
        return nil
    }

    return GetRodents(res.Hits.Hits)
}
```

[Elasticsearch Query DSL | Elastic]

elastic

Result Processing

rodent-processing.go

```go
func GetRodents(hits []types.Hit) []Rodent {
    var rodents []Rodent

    for _, hit := range hits {
        var currentRodent Rodent
        err := json.Unmarshal(hit.Source_, &currentRodent)

        if err != nil {
            log.Fatal(err)
            return nil
        }

        currentRodent.ID = hit.Id_
        rodents = append(rodents, currentRodent)
    }

    return rodents
}
```

elastic

Semantic Search: Meaning, not literal matches

Elasticsearch:
You Know, for Vector Search

Bag of Words

Pocket gophers, commonly referred to simply as gophers, are burrowing rodents of the family Geomyidae.

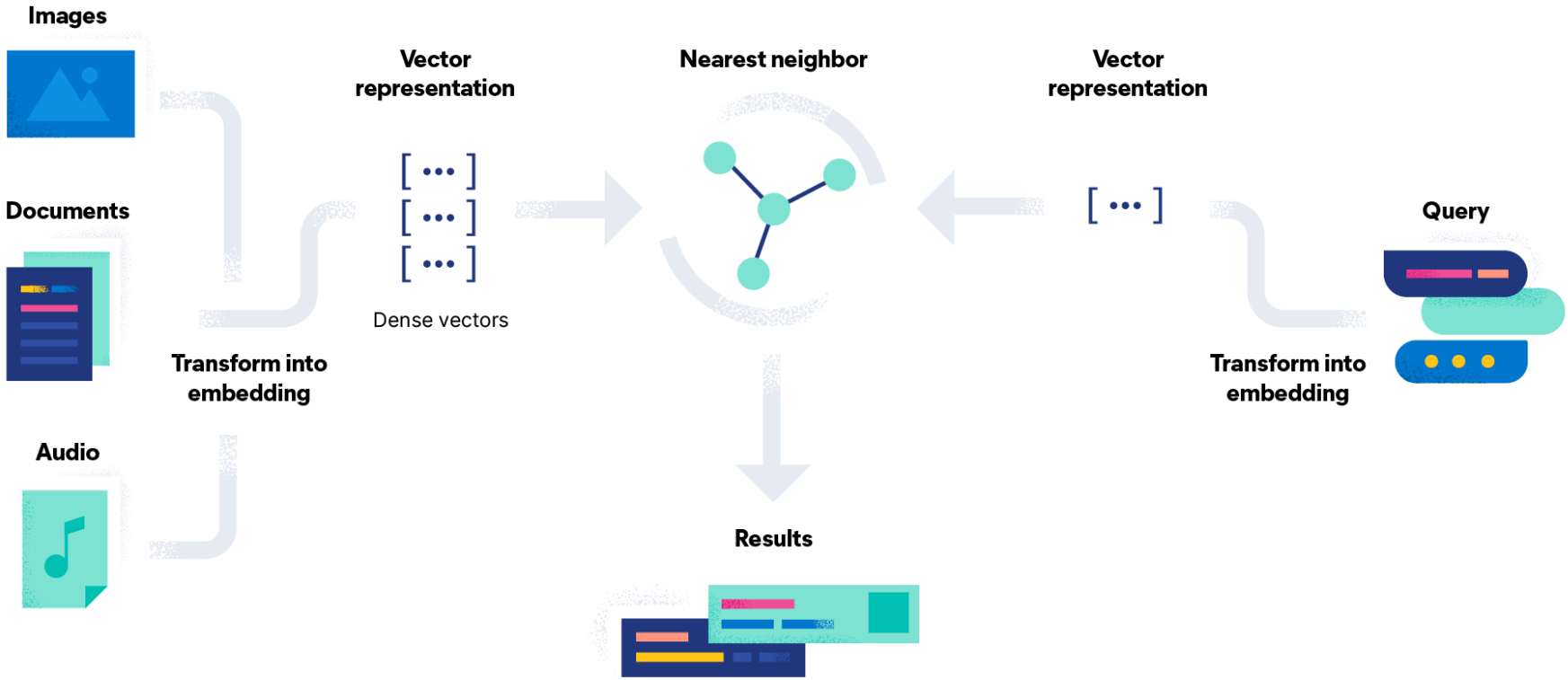
The capybara, or greater capybara, is a giant cavy rodent.

pocket: 1, **gopher: 2**, commonly: 1, referred: 1, simply: 1, burrowing: 1, **rodent: 2**, family: 1, geomyidae: 1, capybara: 2, greater: 1, giant: 1, cavy: 1, native: 1

```
[pocker, gopher, commonly, referred, simply, burrowing, rodent,
family, geomyidae, capybara, greater, giant, cavy, native]
[1,   2,   1,   1,   1,   1,   1,   1,   1,   0,   0,   0,   0,   0]
[0,   0,   0,   0,   0,   0,   1,   0,   0,   2,   1,   1,   1,   1]
```

elastic

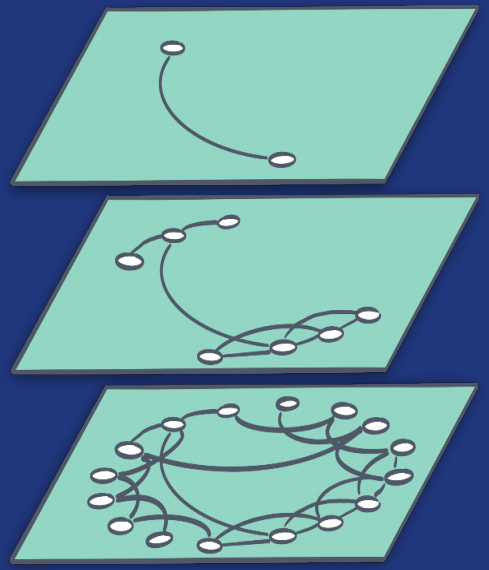
Vector Search



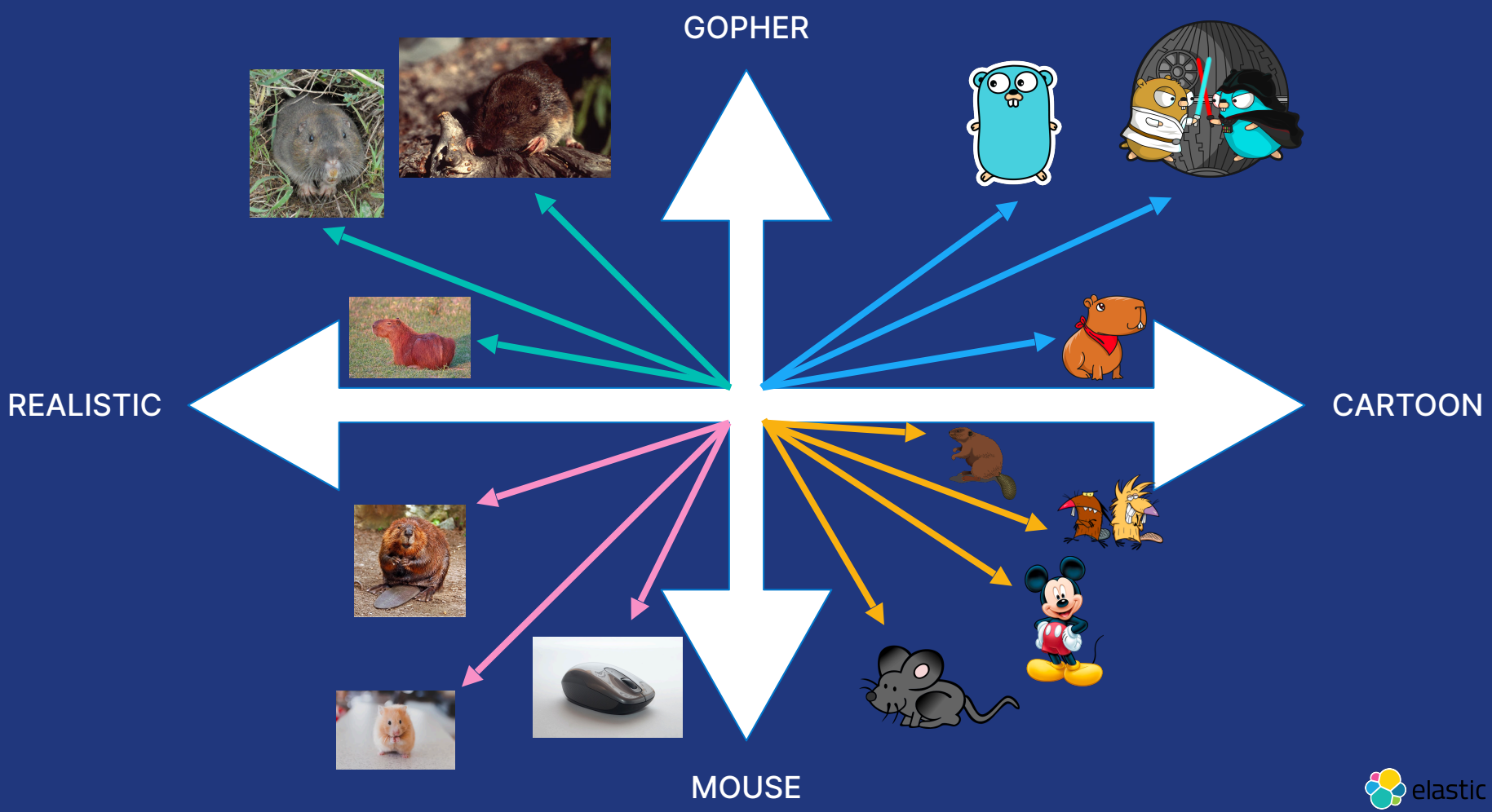
How vectors are indexed for search: graphs

Indexing documents with dense vectors

- **Hierarchical Navigable Small Worlds:** 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

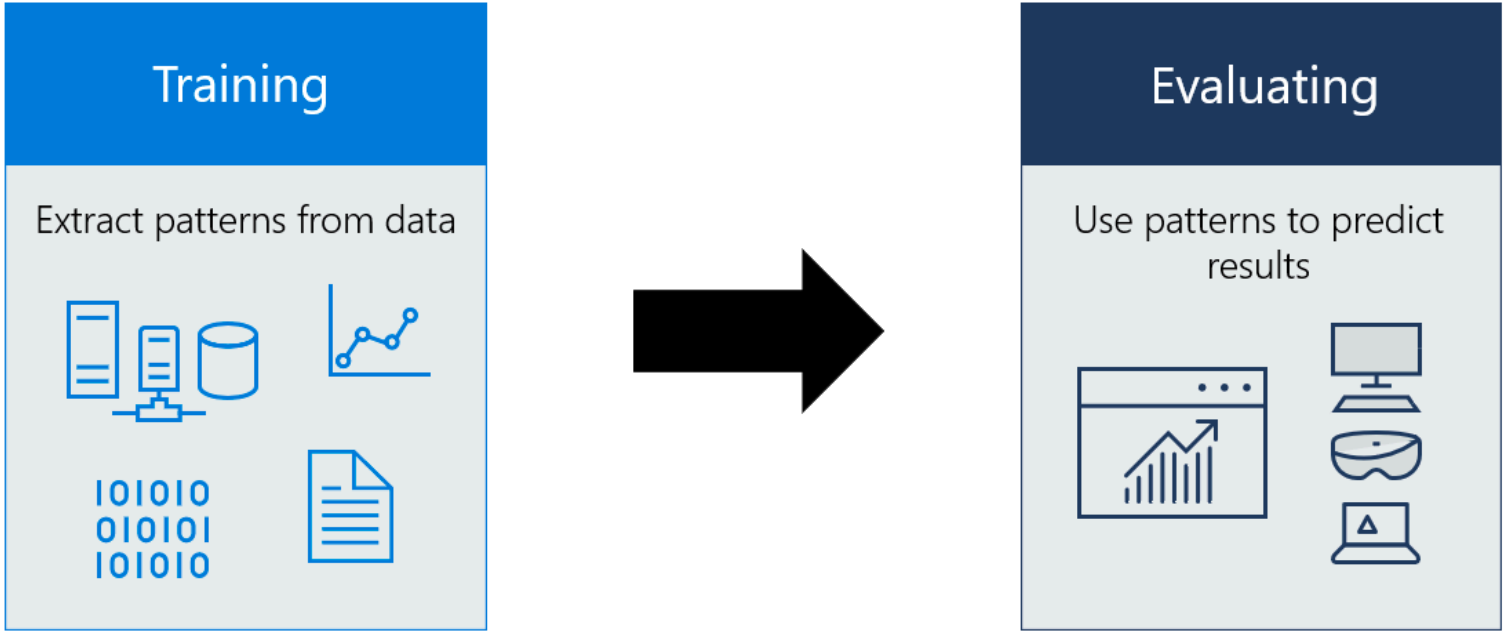


elastic



elastic

What is a model?



elastic

Free vs Paid (Platinum)

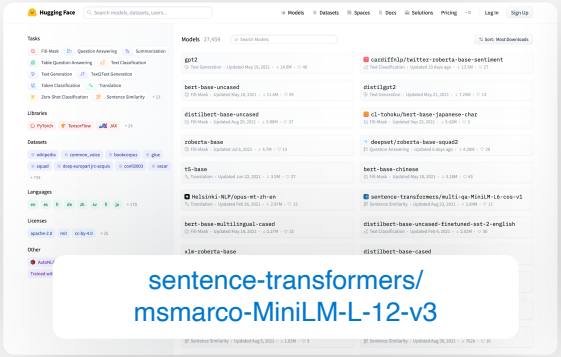
kNN with HNSW

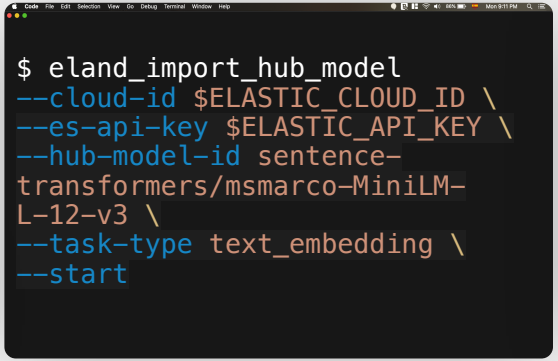
1. Bring your own embeddings
2. Enough memory for vectors to fit in off-heap RAM

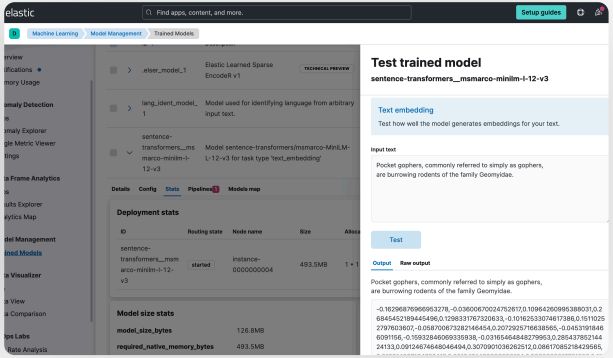
Inference in Elasticsearch

1. Generate embeddings (like HuggingFace transformers)
2. ELSER

Setting Up the Embedding Model







PyTorch → eland →

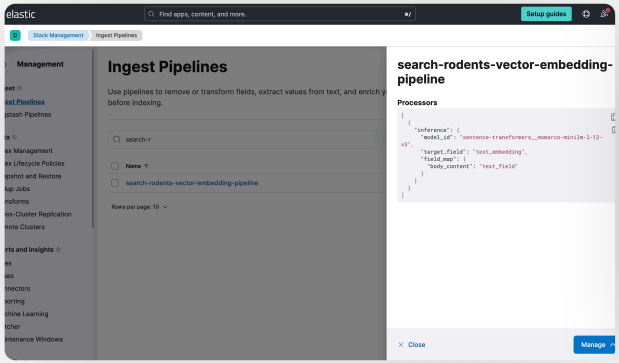
Select the appropriate model

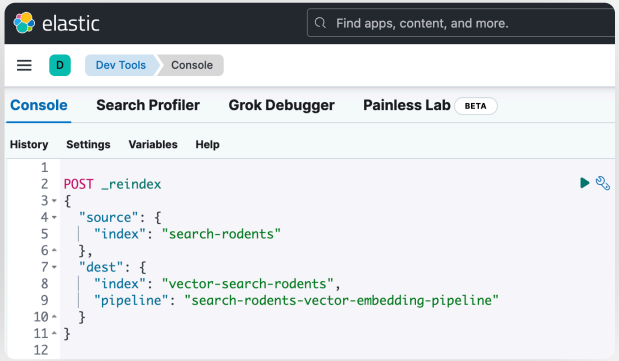
Load the model to the cluster

Manage models

elastic

Vector Document





PyTorch

Enrich data using inference processor

Reindex data using ingest pipeline

Generate embeddings

[Ingest pipeline Inference processor | Elastic]

elastic

Vector Search

vector-search-filtered.go

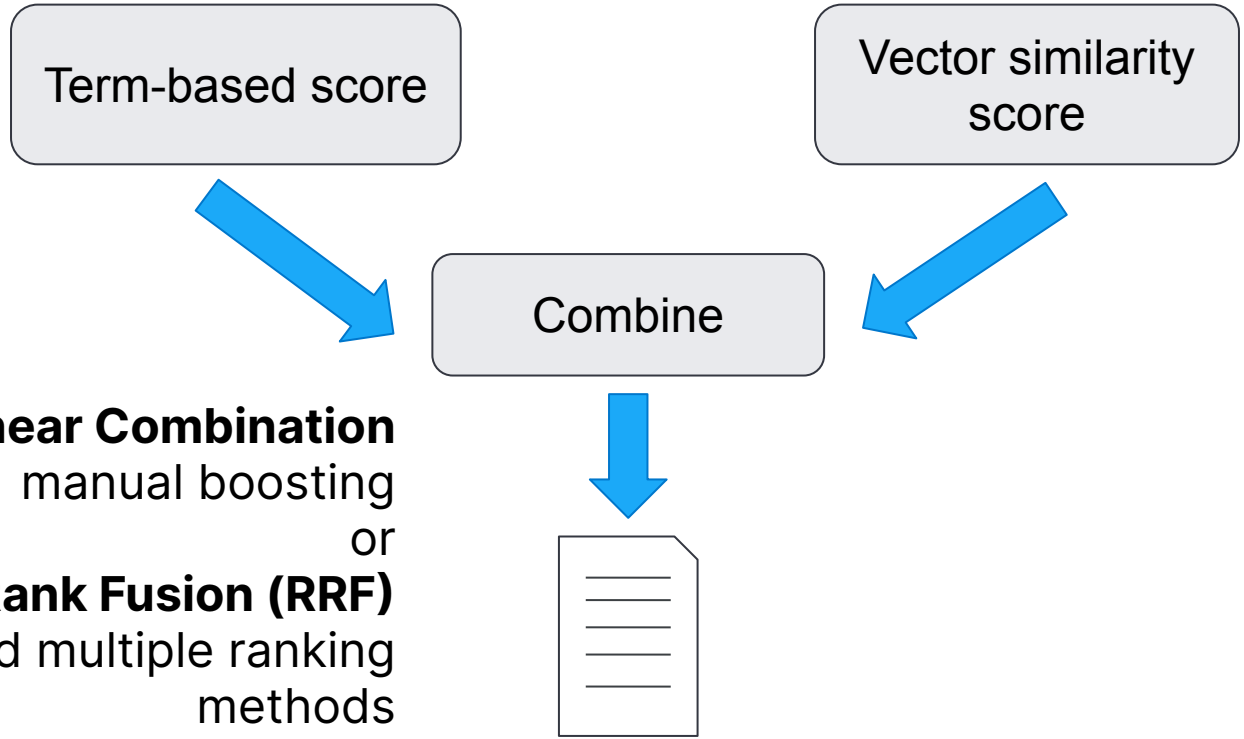
```go
// Vector search example with filter
func VectorSearchWithFilter(term string) []Rodent {
    res, err := client.Search().
        Index("vector-search-rodents").
        Knn(types.KnnQuery{
            Field: "text_embedding.predicted_value",
            K:     10,
            Filter: []types.Query{
                {
                    Match: map[string]types.MatchQuery{
                        "body_content": {Query: "rodent"},
                    },
                },
            },
            NumCandidates: 10,
            QueryVectorBuilder: &types.QueryVectorBuilder{
                TextEmbedding: &types.TextEmbedding{
                    ModelId:   "sentence-transformers__msmarco-minilm-l-12-v3",
                    ModelText: term,
                },
            }}).Do(context.Background())

    if err != nil {
        log.Fatal(err)
        return nil
    }

    return GetRodents(res.Hits.Hits)
}
```

[Elasticsearch kNN search | Elastic]

elastic

Elasticsearch:
You Know, for Hybrid Search

Hybrid Scoring



Linear Combination
manual boosting
or
Reciprocal Rank Fusion (RRF)
blend multiple ranking methods

Manual Boosting

hybrid-search-manual.go

```go
// Hybrid search example
func HybridSearchWithBoost(term string) []Rodent {
    var knnBoost float32 = 0.2

    res, err := client.Search().
        Index("vector-search-rodents").
        Knn(types.KnnQuery{
            Field:         "text_embedding.predicted_value",
            Boost:         &knnBoost,
            K:             10,
            NumCandidates: 10,
            QueryVectorBuilder: &types.QueryVectorBuilder{
                TextEmbedding: &types.TextEmbedding{
                    ModelId:   "sentence-transformers__msmarco-minilm-l-12-v3",
                    ModelText: term,
                },
            }}).
        Query(&types.Query{
            Match: map[string]types.MatchQuery{
                "title": {Query: term},
            },
        }).
        From(0).
        Size(2).
        Do(context.Background())

    if err != nil {
        log.Fatal(err)
        return nil
    }

    return GetRodents(res.Hits.Hits)
}
```

[Elasticsearch search multiple kNN fields | Elastic]

elastic

Reciprocal Rank Fusion

rrf.py

```
score = 0.0
for q in queries:
    if d in result(q):
        score += 1.0 / ( k + rank( result(q), d ) )
return score

# where
# k is a ranking constant
# q is a query in the set of queries
# d is a document in the result set of q
# result(q) is the result set of q
# rank( result(q), d ) is d's rank within the result(q)
starting from 1
```

[Elasticsearch Reciprocal Rank Fusion | Elastic]

elastic

Reciprocal Rank Fusion

hybrid-search-rrf.go

```go
// Hybrid search with RRF
func HybridSearchWithRRF(term string) []Rodent {
    var windowSize int64 = 10 // min required
    var rankConstant int64 = 42

    res, err := client.Search().
        Index("vector-search-rodents").
        Knn(types.KnnQuery{
            Field:         "text_embedding.predicted_value",
            K:             10,
            NumCandidates: 10,
            QueryVectorBuilder: &types.QueryVectorBuilder{
                TextEmbedding: &types.TextEmbedding{
                    ModelId:   "sentence-transformers__msmarco-minilm-l-12-v3",
                    ModelText: term,
                },
            }}).
        Query(&types.Query{
            Match: map[string]types.MatchQuery{
                "title": {Query: term},
            },
        }).
        Rank(&types.RankContainer{
            Rrf: &types.RrfRank{
                WindowSize:   &windowSize,
                RankConstant: &rankConstant,
            },
        }).
        Do(context.Background())

    if err != nil {
        log.Fatal(err)
        return nil
    }

    return GetRodents(res.Hits.Hits)
}
```

[Elasticsearch Reciprocal Rank Fusion | Elastic]

elastic

Thank you!

SCAN ME

Carly Richmond
@CarlyLRichmond