

Multi-Stage Retrieval in Elasticsearch

Feb 2026

Carlos Delgado

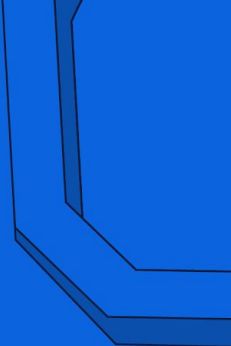
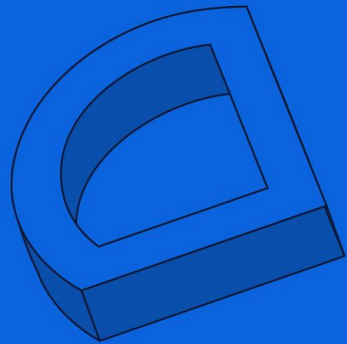
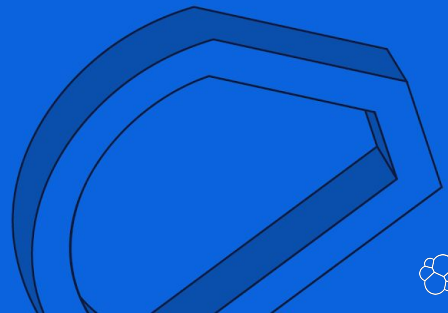


elastic

| The Search
AI Company

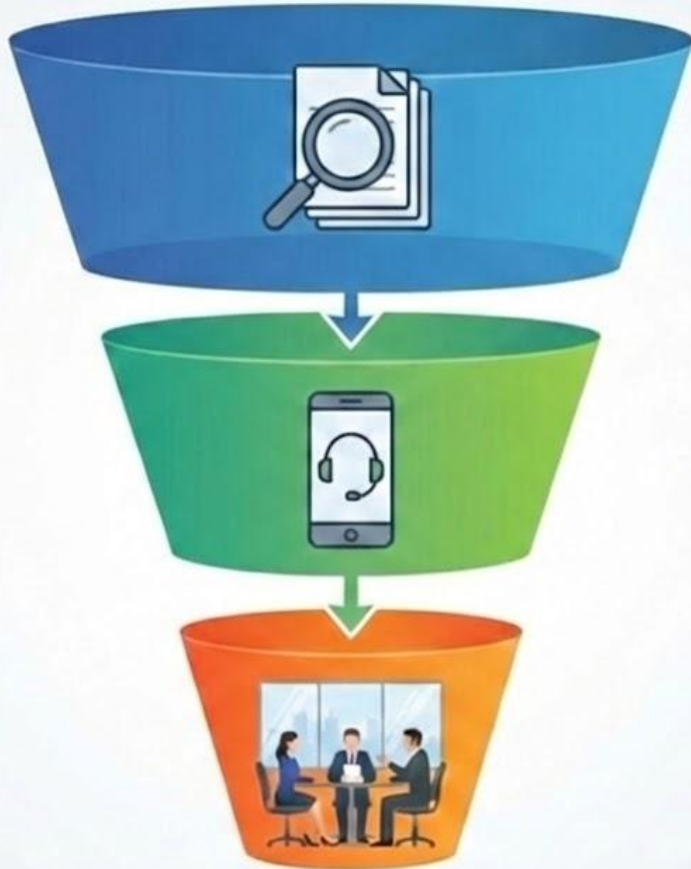


Why do we need multi stage retrieval?



An analogy: The interview process

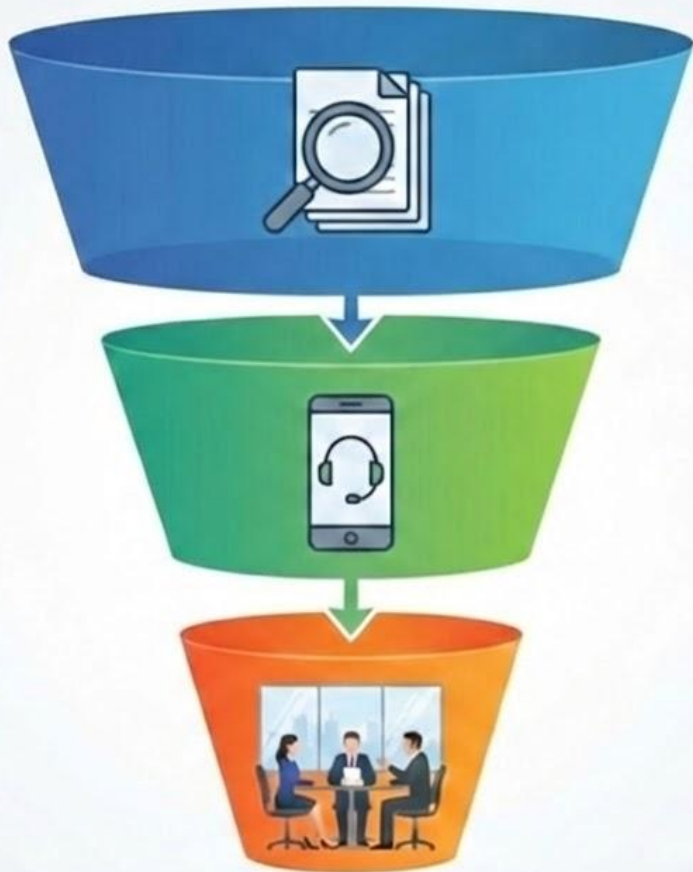




Resume scan

Phone screen call

Interviews



Resume scan
(many)

Phone screen call
(several)

Interviews
(few)



Resume scan
(seconds)

Phone screen call
(minutes)

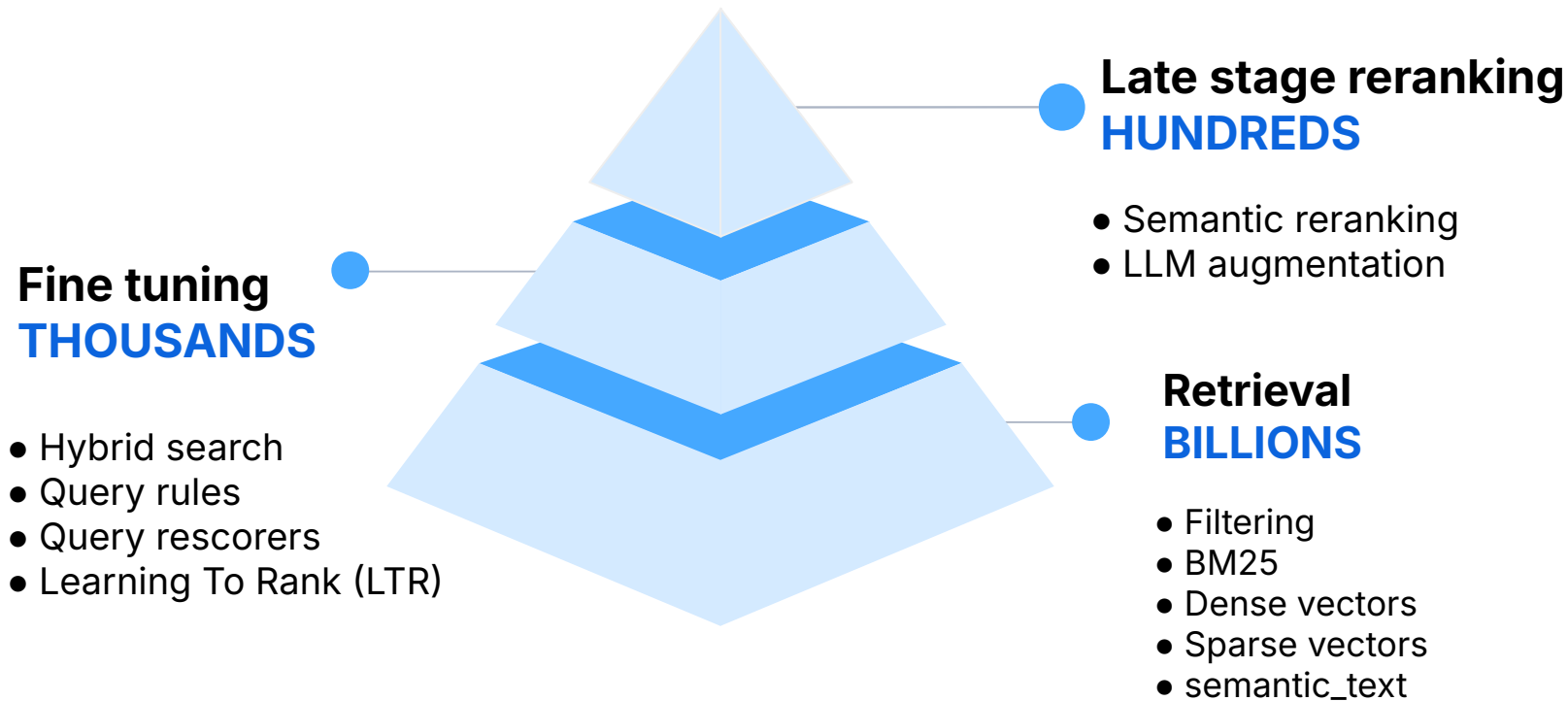
Interviews
(hours)

The challenge

Why do we need multi stage retrieval?

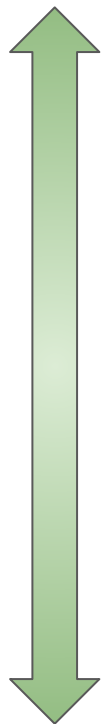
- Massive candidate space
 - Millions to billions of pages, products, users...
- Latency expectations
 - Users expect results in hundreds of milliseconds
- More expensive relevance models
 - Neural re-rankers, cross-encoders, personalization, LLMs are too slow to run on everything
- Quality vs. cost trade-off
 - Simple models → fast but less accurate
 - Complex models → accurate but slow & expensive

Multi-Stage Retrieval



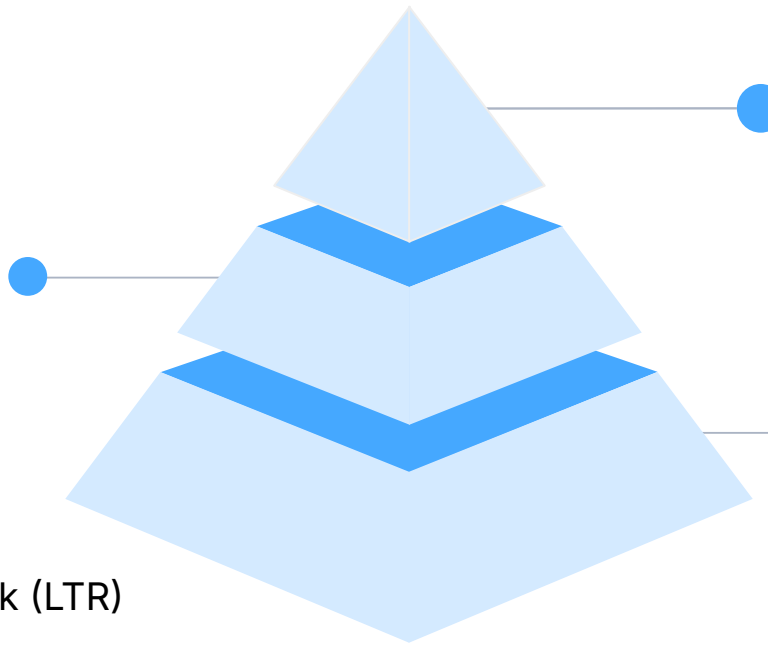
Multi-Stage Retrieval

Precision



Fine tuning THOUSANDS

- Hybrid search
- Query rules
- Query rescorers
- Learning To Rank (LTR)



Late stage reranking HUNDREDS

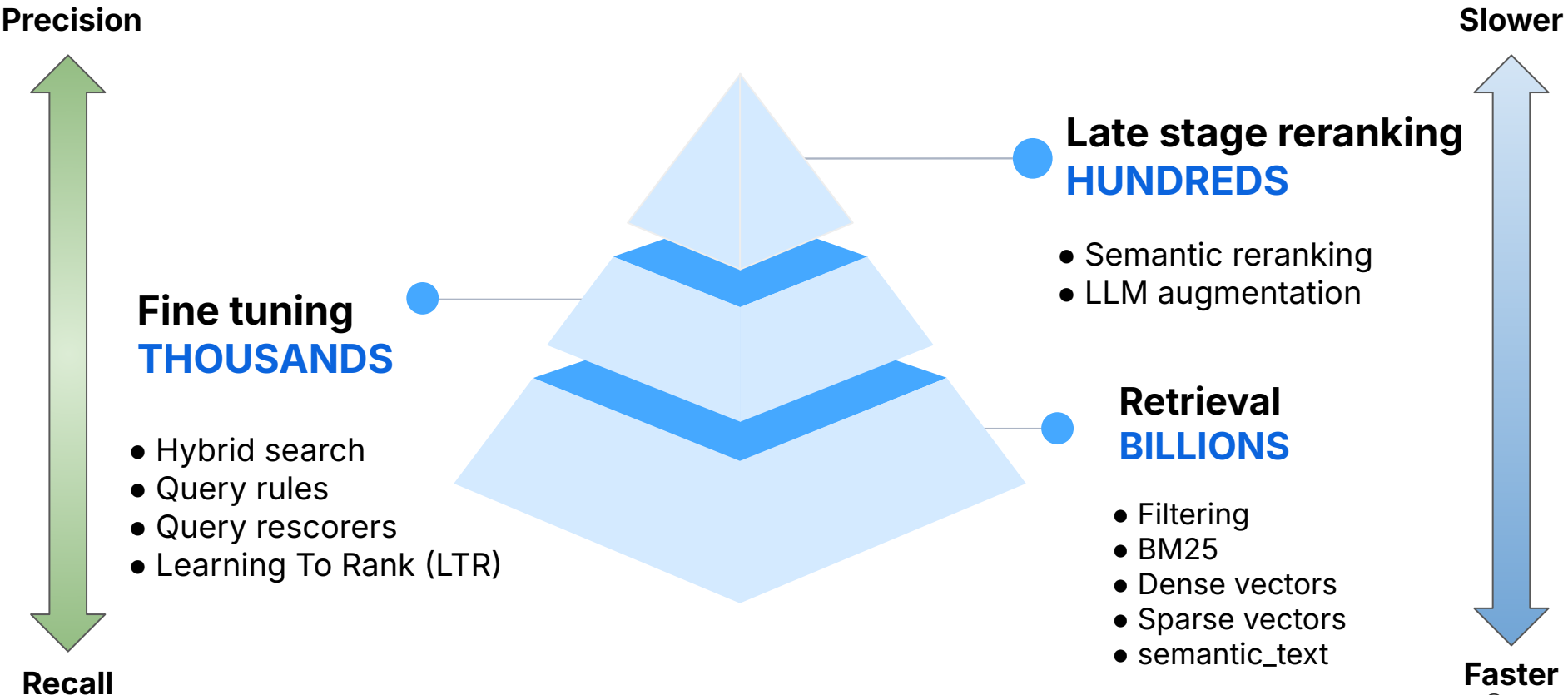
- Semantic reranking
- LLM augmentation

Retrieval BILLIONS

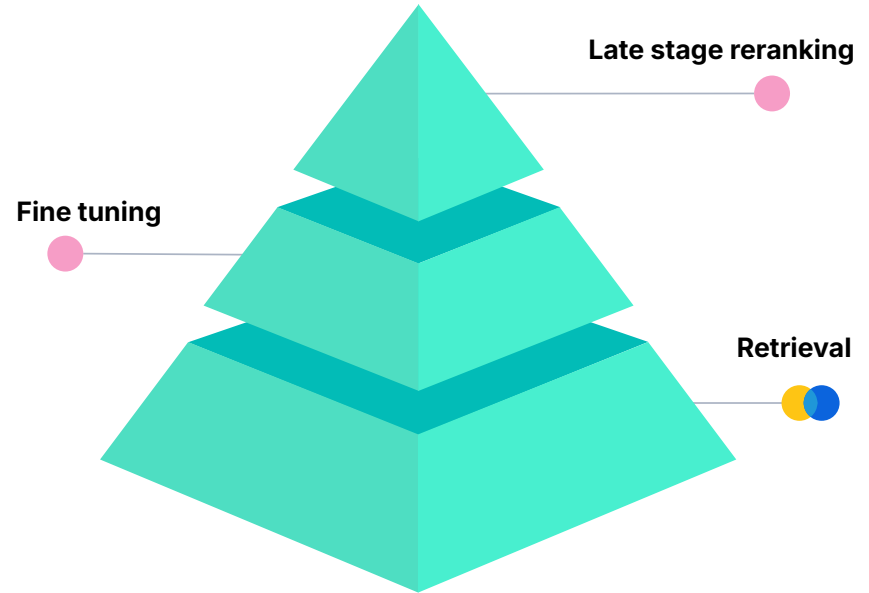
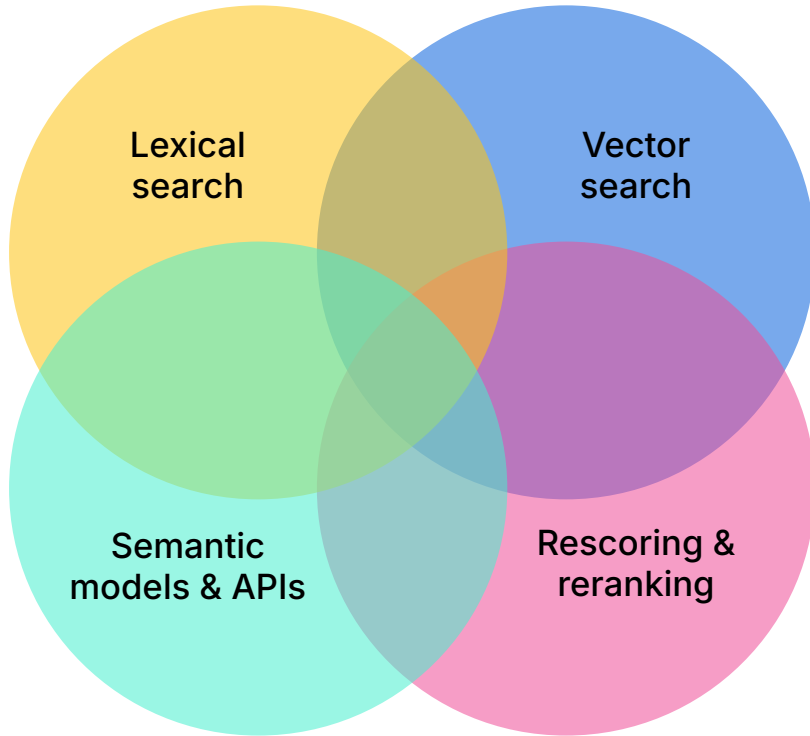
- Filtering
- BM25
- Dense vectors
- Sparse vectors
- semantic_text

Recall

Multi-Stage Retrieval



Putting it all together



Search as a language

The Query DSL

- Tree structure, based around a query (match, knn, term...)
- Query can be composed of other queries
 - Combine them (bool, dis_max)
 - Change the results (script_score, function_score, boost)
- Retrievers are used to wrap other queries
 - Single query (standard, knn)
 - Combine them (rrf, linear)
 - Change their results (rerank, rescore, pinned, rule)

```
{
  "size": 10,
  "retriever": {
    "text_similarity_reranker": {
      "field": "text",
      "inference_id":
      "cohere-rerank-v3-model",
      "inference_text": "{{query}}",
      "min_score": 0.6,
      "standard": {
        "query": {
          "bool": {
            "should": [
              {
                "multi_match": {
                  "query": "{{query}}",
                  "fields": ["title^2",
"body"]
                }
              },
              {
                "semantic": {
                  "field": "semantic_text",
                  "query": "{{query}}"
                }
              }
            ]
          }
        }
      },
      "rescore": {
        "query": {
          "rescore_query": {
            "script_score": {
              "script": {
                "source":
"decayDateGauss('now', '30d', '7d', 0.5',
doc['publish_date'].value.toInstant().toEpochM
illi())"
              }
            }
          }
        }
      }
    }
  }
}
```

ES|QL

- ES|QL is a piped query language for filtering, transforming, and analyzing data.
- Each command is a step in a processing pipeline
- Very similar to a funnel or a factory

```
FROM catalog
| WHERE category == "Clothing" AND price < 100
| STATS count_by_supplier = COUNT(*) BY supplier
| SORT count_by_supplier DESC
| LIMIT 20
```

Why ES|QL?

Processing model

- Processing model for ES|QL:
 - Start with all your data
 - Process it on steps
 - Output for each step is the input for the next one

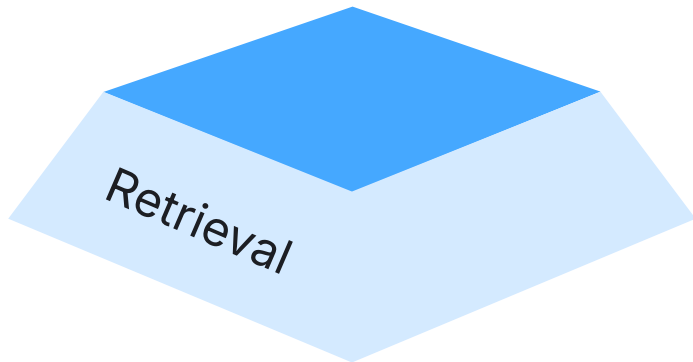
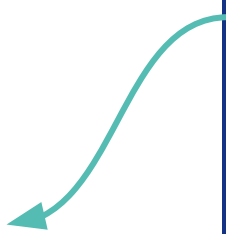
- This model is useful for multi-stage retrieval
 - Start considering all results
 - Filter on each step
 - Each step can be more expensive to apply, as it processes a reduced subset of the data

Multi-Stage Retrieval with ES|QL

Basic Query

Data origin

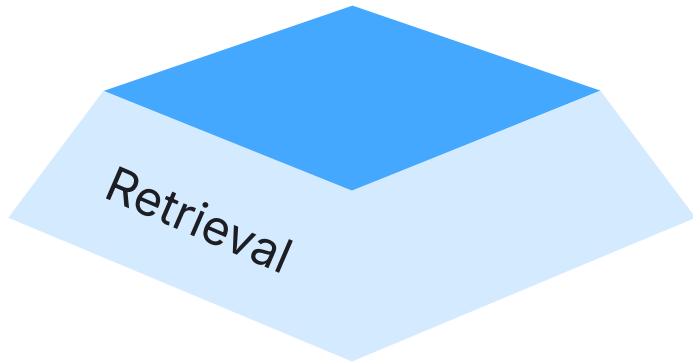
FROM catalog



Basic Query

Basic filters

```
FROM catalog  
| WHERE category = "Clothing" AND price < 100
```



Basic Query

Calculate relevancy
adding `_score` column

Basic BM25 match
- updates `_score`

```
FROM catalog METADATA _score  
| WHERE category = "Clothing" AND price < 100  
| WHERE MATCH(product, ?query)
```

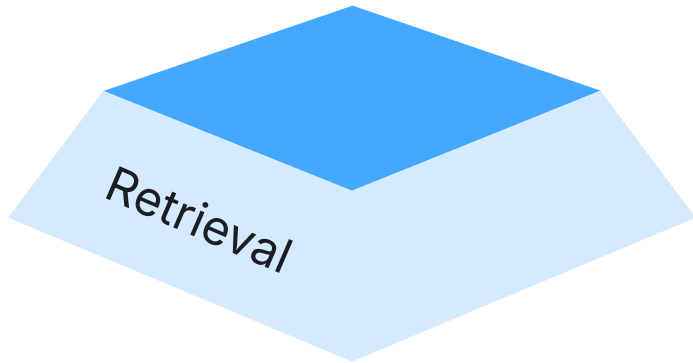


Retrieval

Basic Query

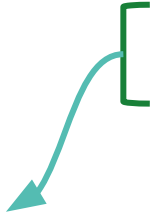
```
FROM catalog METADATA _score  
| WHERE category = "Clothing" AND price < 100  
| WHERE MATCH(product, ?query)  
| SORT _score DESC  
| LIMIT 10
```

Top 10 relevant results

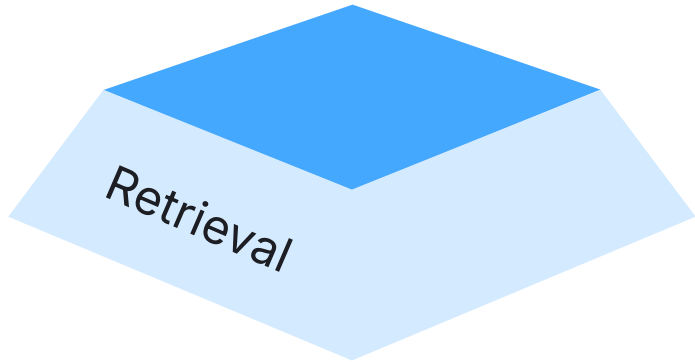


Combining

Basic boost combination



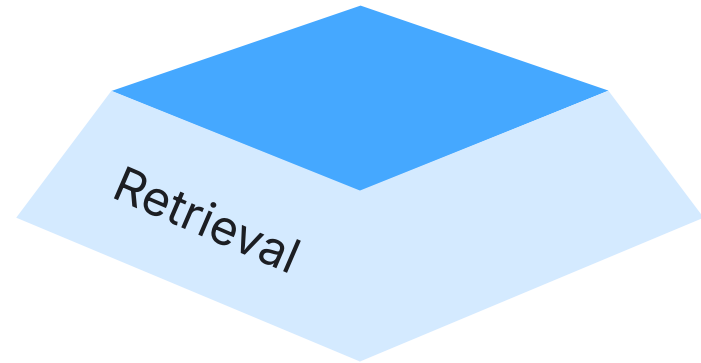
```
FROM catalog METADATA _score
| WHERE category = "Clothing" AND price < 100
| WHERE MATCH(product, ?query, {"boost": 3.0}) OR
      MATCH(description, ?query)
| SORT _score DESC
| LIMIT 10
```



Combining

Include a semantic query
- updates _score

```
FROM catalog METADATA _score
| WHERE category = "Clothing" AND price < 100
| FORK
    (MATCH(product, ?query, {"boost": 3.0}) OR
     MATCH(description, ?query))
    (MATCH(product_semantic, ?query))
| FUSE RRF
| SORT _score DESC
| LIMIT 10
```



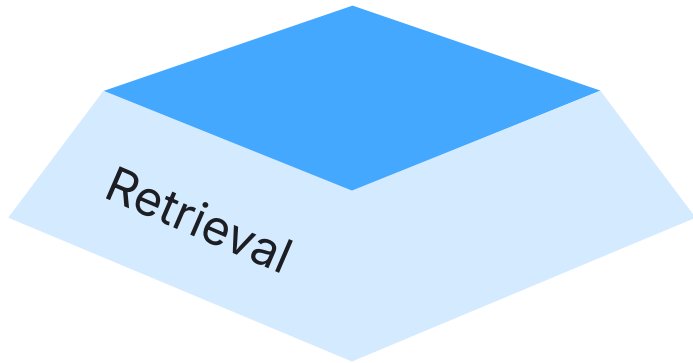
Combining

Lexical query

Semantic query

Combine using RRF...

```
FROM catalog METADATA _score
| WHERE category = "Clothing" AND price < 100
| FORK
|   (MATCH(product, ?query, {"boost": 3.0}) OR
|     MATCH(description, ?query))
|   (MATCH(product_semantic, ?query))
| FUSE RRF
| SORT _score DESC
| LIMIT 10
```



Combining

Lexical query

Semantic query

... or linear combination

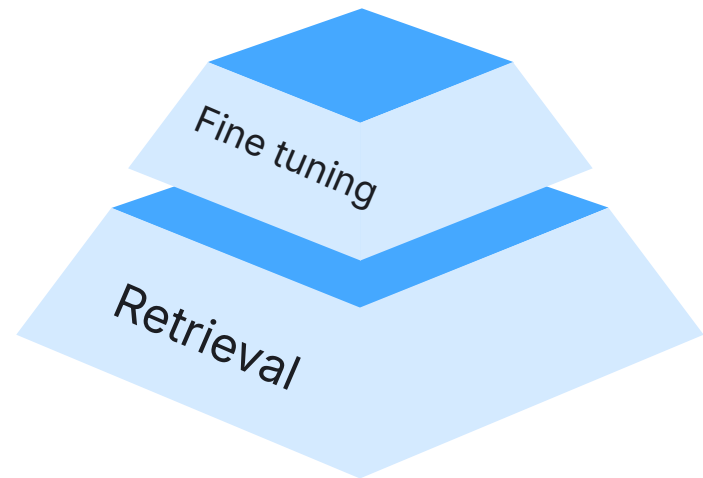
```
FROM catalog METADATA _score
| WHERE category = "Clothing" AND price < 100
| FORK
  (MATCH(product, ?query, {"boost": 3.0}) OR
   MATCH(description, ?query))
  (MATCH(product_semantic, ?query))
| FUSE LINEAR
  WITH { "weights": { "fork1": 0.7, "fork2": 0.3 },
        "normalizer": "minmax" }
| SORT _score DESC
| LIMIT 10
```



Retrieval

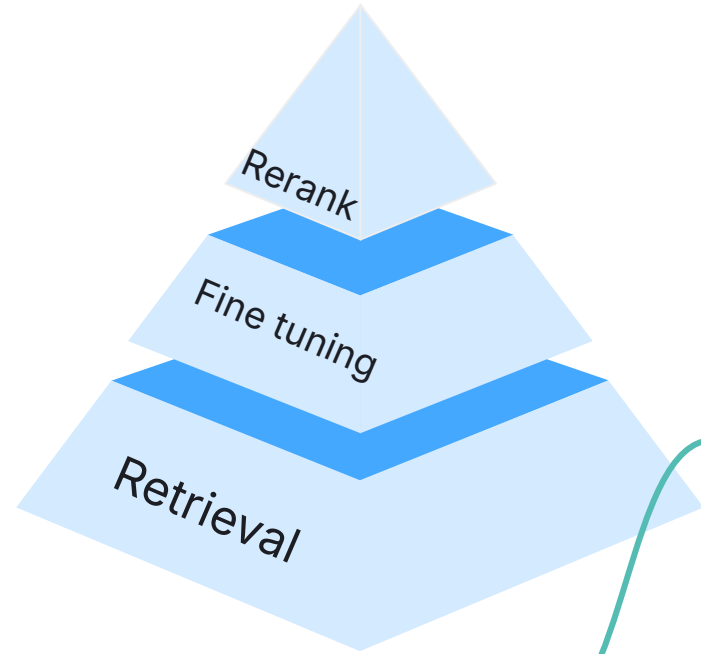
Rescoring

Modify the original score
with other signals



```
FROM catalog METADATA _score
| WHERE category = "Clothing" AND price < 100
| FORK
  (MATCH(product, ?query, {"boost": 3.0}) OR
   MATCH(description, ?query))
  (MATCH(product_semantic, ?query))
| FUSE RRF
| EVAL date_score = DECAY(date_added, NOW(), 7 days)
| EVAL combined_score = _score + date_score + sponsored
| SORT combined_score DESC
| LIMIT 30
```

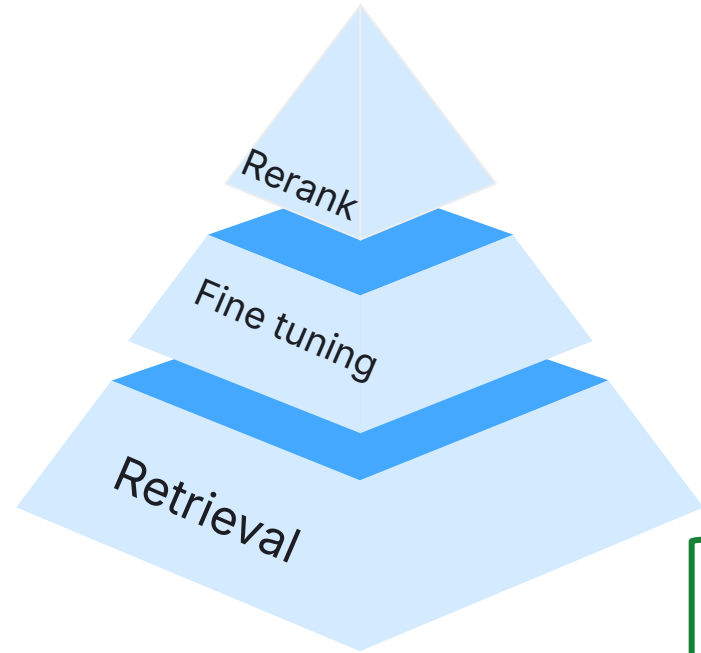
Reranking



Semantic rerank on the previous top 30 results

```
FROM catalog METADATA _score
| WHERE category = "Clothing" AND price < 100
| FORK
  (MATCH(product, ?query, {"boost": 3.0}) OR
   MATCH(description, ?query))
  (MATCH(product_semantic, ?query))
| FUSE RRF
| EVAL date_score = DECAY(date_added, NOW(), 7 days)
| EVAL combined_score = _score + date_score + sponsored
| SORT combined_score DESC
| LIMIT 30
| RERANK ?query ON product, description
  WITH {"inference_id": "jina-reranker-v3"}
| SORT _score DESC
| LIMIT 5
```

Use your LLMs



Create a prompt and use an LLM on the results

```
FROM catalog METADATA _score
| WHERE category = "Clothing" AND price < 100
| FORK
  (MATCH(product, ?query, {"boost": 3.0}) OR
   MATCH(description, ?query))
  (MATCH(product_semantic, ?query))
| FUSE RRF
| EVAL date_score = DECAY(date_added, NOW(), 7 days)
| EVAL combined_score = _score + date_score + sponsored
| SORT combined_score DESC
| LIMIT 30
| RERANK ?query ON product, description
  WITH {"inference_id": "jina-reranker-v3"}
| SORT _score DESC
| LIMIT 5
| EVAL prompt = CONCAT("Summarize this product and what
  are the best use cases for it: \n",
  "Product name: ", product, "\n",
  "Description: ", description, "\n")
| COMPLETION summary = prompt
  WITH { "inference_id" : "chatgpt-5.2" }
```

Combining
results

Filtering

Lexical search
Semantic search

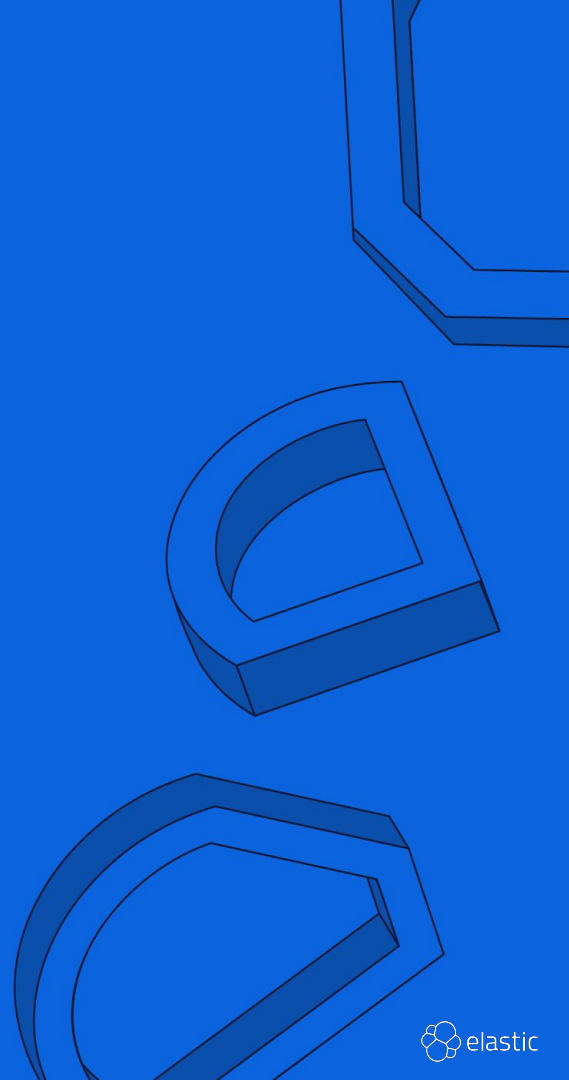
Rescoring

Reranking

LLM Augmentation

```
FROM catalog METADATA _score
| WHERE category = "Clothing" AND price < 100
| FORK
  (MATCH(product, ?query, {"boost": 3.0}) OR
   MATCH(description, ?query))
  (MATCH(product_semantic, ?query))
| FUSE RRF
| EVAL date_score = DECAY(date_added, NOW(), 7 days)
| EVAL combined_score = _score + date_score + sponsored
| SORT combined_score DESC
| LIMIT 30
| RERANK ?query ON product, description
  WITH {"inference_id": "jina-reranker-v3"}
| SORT _score DESC
| LIMIT 5
| EVAL prompt = CONCAT("Summarize this product and what
  are the best use cases for it: \n",
  "Product name: ", product, "\n",
  "Description: ", description, "\n")
| COMPLETION summary = prompt
  WITH { "inference_id" : "chatgpt-5.2" }
```

Coming Soon



Coming soon

Improving search support in ES|QL

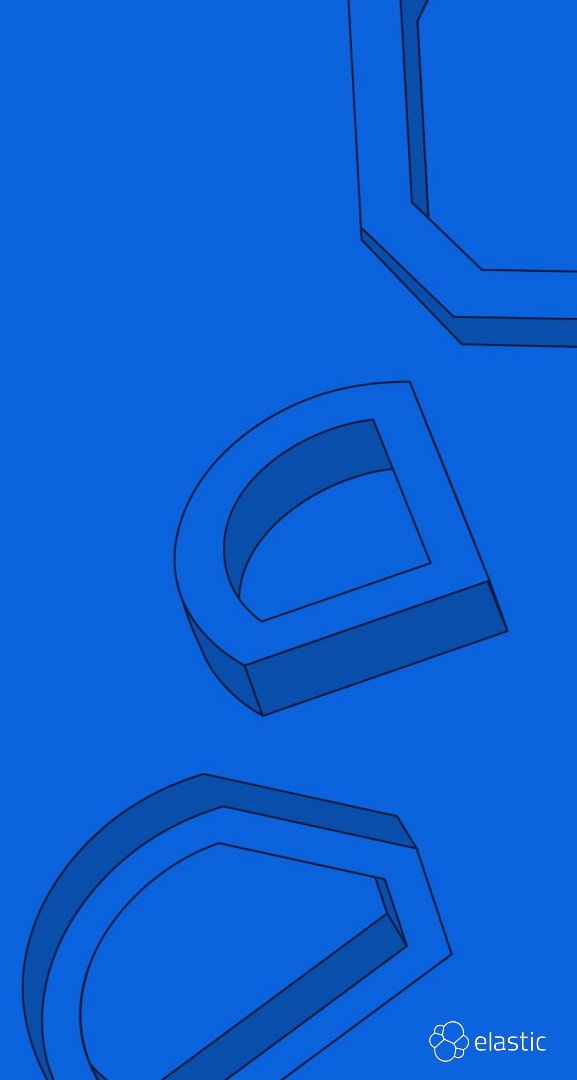
- Sparse vector model support
 - `sparse_vector` data type
 - `semantic_text` with `sparse_vector`
- Highlighting
- Vector arithmetic
 - Custom scoring and vector manipulation
- Semantic search
 - Cross Cluster Search support (9.3)

Coming soon

Not just porting, better search

- Results diversification
 - MMR command (Maximum Marginal Relevance)
- Multimodal search
 - Image support

Conclusions



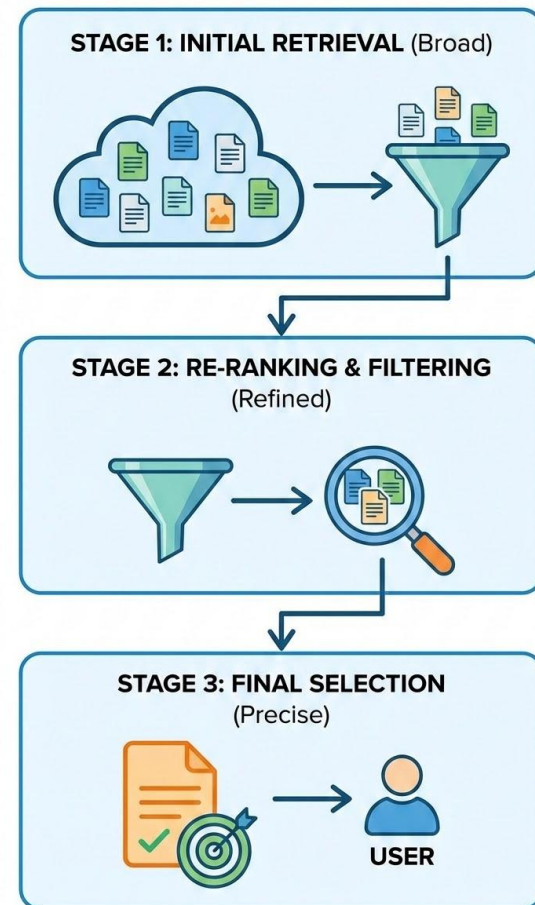
Conclusions

Multi stage retrieval and ES|QL

Multi-stage retrieval allows using better ranking methods by

- Casting a wide net first (fast, cheap)
- Gradually filtering and refining results
- Running expensive ranking only on top candidates

MULTI-STAGE RETRIEVAL



Conclusions

Multi stage retrieval and ES|QL

ES|QL helps building multi stage steps in a single and simple language

- Piped query syntax makes processing steps explicit
- Supports hybrid search (lexical + vector + fusion)
- Combine retrieval, filtering, aggregation, and custom scoring without switching languages or layers

The background is a solid blue color. It features several large, stylized, 3D-rendered geometric shapes in a lighter shade of blue. These shapes are irregular polygons with beveled edges, resembling architectural elements or abstract architectural forms. They are positioned in the corners and along the sides of the frame, creating a modern, geometric aesthetic.

Questions?