Executing Search Requests Using Elasticsearch Query DSL

Overview

Introduce the Query DSL that Elasticsearch uses for search queries

Understand the contexts in which search queries run

Work with full text searches, term searches, compound searches, filters

Understand how relevance is calculated

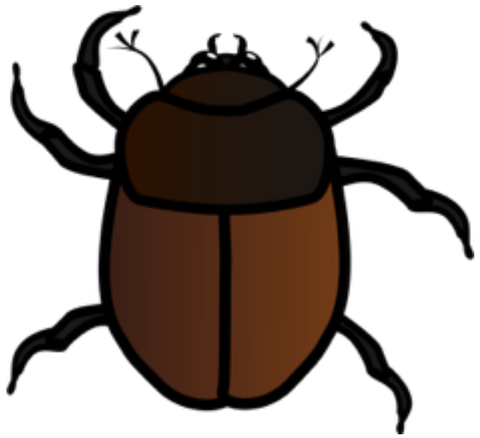
Implement all queries using the Elasticsearch REST API

Recap: How Does Search Work?

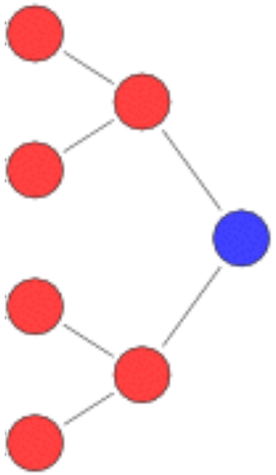
What Is the Objective of Search?

Find the most relevant documents with your search terms

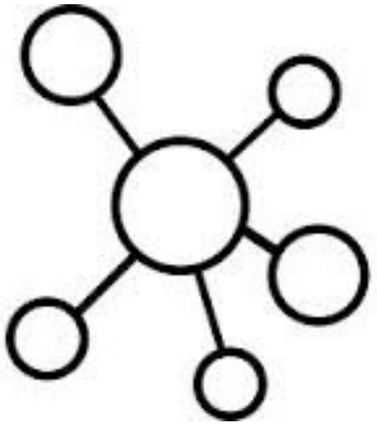
Most Relevant Document for Search Terms



Know of the document's existence



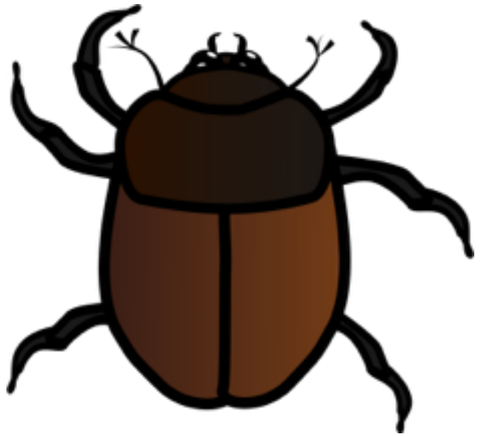
Index the document for lookup



Know how relevant the document is

Retrieve ranked by relevance

Most Relevant Document for Search Terms



Web crawler

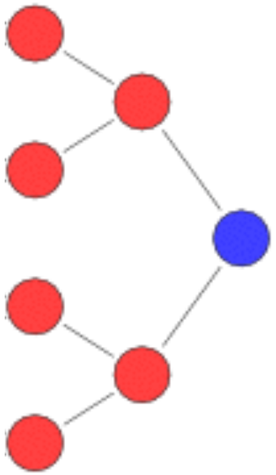
Index the document for lookup

Know how relevant the document is

Retrieve ranked by relevance

Most Relevant Document for Search Terms

Web crawler



Inverted index

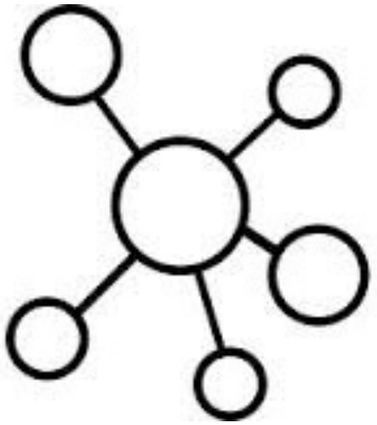
Know how relevant the document is

Retrieve ranked by relevance

Most Relevant Document for Search Terms

Web crawler

Inverted index



Scoring

Retrieve ranked by relevance

Most Relevant Document for Search Terms

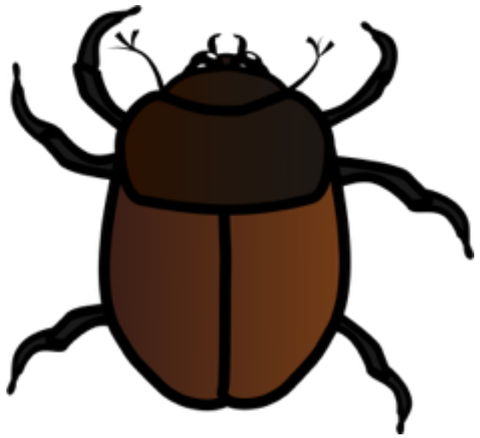
Web crawler

Inverted index

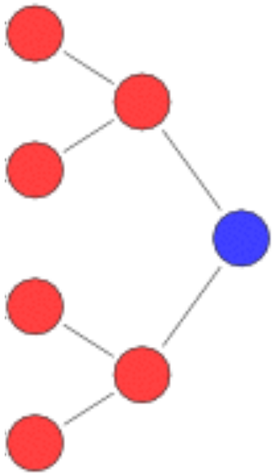
Scoring

Search

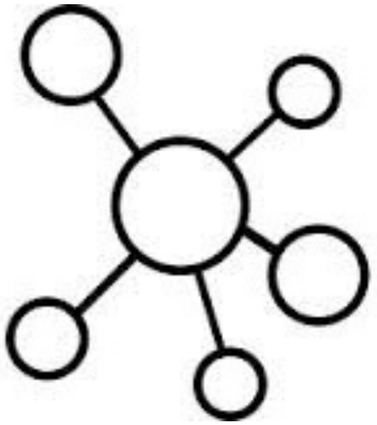
Most Relevant Document for Search Terms



Web crawler



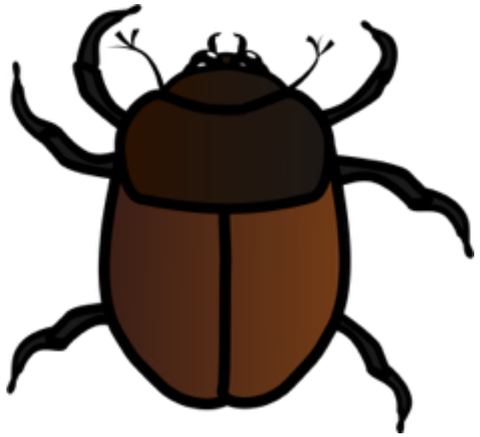
Inverted index



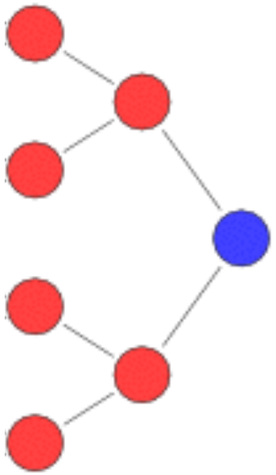
Scoring

Search

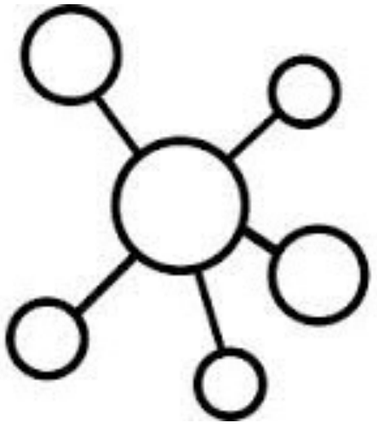
Elasticsearch



Specify documents to index



Elasticsearch creates the inverted index behind the scenes



Applies a scoring algorithm for each document for each term

Retrieves documents using a Query DSL

The Query DSL

Query DSL

A flexible, expressive search language that Elasticsearch uses to expose most of the power of Lucene through a simple JSON interface. It is what you should be using to write your queries in production. It makes your queries more flexible, more precise, easier to read, and easier to debug.

https://www.elastic.co/guide/en/elasticsearch/guide/current/query-dsl-intro.html

Query DSL

A flexible, expressive search language that Elasticsearch uses to expose most of the power of Lucene through a simple JSON interface. It is what you should be using to write your queries in production. It makes your queries more flexible, more precise, easier to read, and easier to debug.

https://www.elastic.co/guide/en/elasticsearch/guide/current/query-dsl-intro.html

Query DSL

A flexible, expressive search language that Elasticsearch uses to expose most of the power of Lucene through a simple JSON interface. It is what you should be using to write your queries in production. It makes your queries more flexible, more precise, easier to read, and easier to debug.

https://www.elastic.co/guide/en/elasticsearch/guide/current/query-dsl-intro.html

Query DSL

A flexible, expressive search language that Elasticsearch uses to expose most of the power of Lucene through a simple JSON interface. It is what you should be using to write your queries in production. It makes your queries more flexible, more precise, easier to read, and easier to debug.

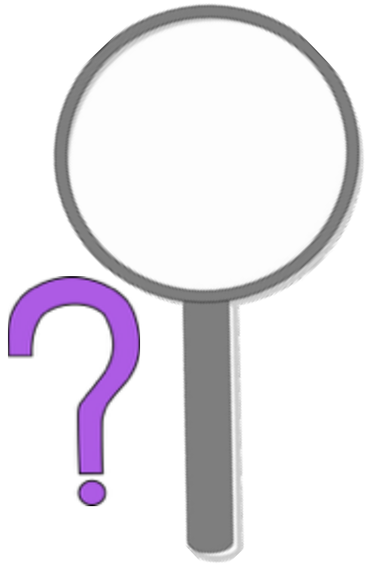
https://www.elastic.co/guide/en/elasticsearch/guide/current/query-dsl-intro.html

Query DSL

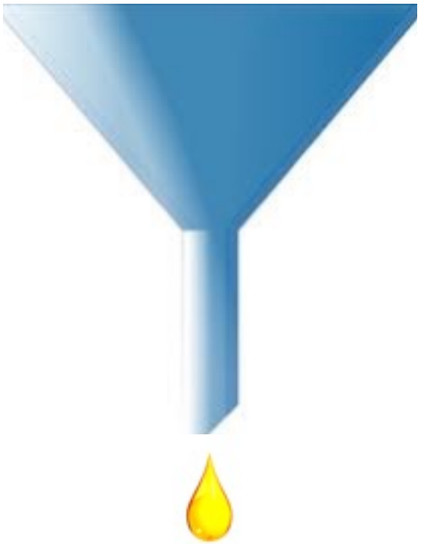
A flexible, expressive search language that Elasticsearch uses to expose most of the power of Lucene through a simple JSON interface. It is what you should be using to write your queries in production. It makes your queries more flexible, more precise, easier to read, and easier to debug.

https://www.elastic.co/guide/en/elasticsearch/guide/current/query-dsl-intro.html

Two Contexts of Search



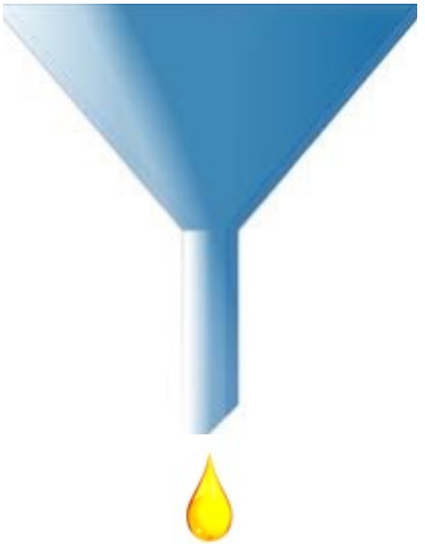
How well does this document match this query?



Does this document match this query clause?

Two Contexts of Search

Query Context



Filter Context

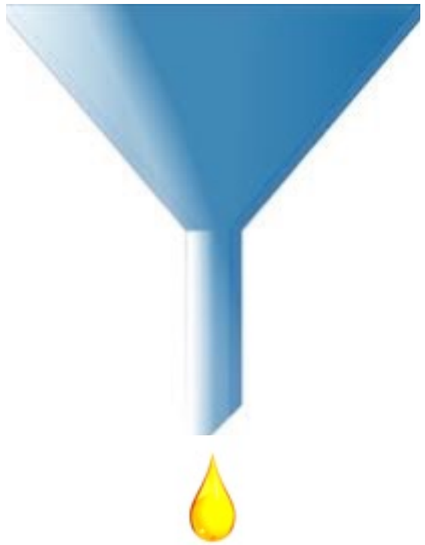
Query Context

Included or not: Determine whether the document should be part of the result

Relevance score: Calculated for every search term the document maps to

High score, more relevant: More relevant documents, higher in the search rankings

Filter Context



Included or not: Yes/no determines whether included in the result

No scoring: No additional relevance ranking in the search results

Structured data: Exact matches, range queries

Faster: Only determine inclusion in results, no scoring to consider

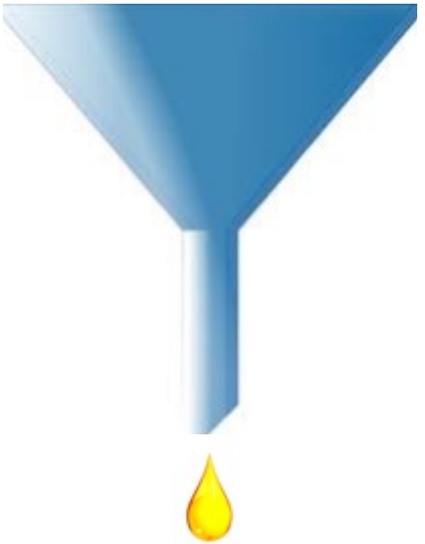
Demo

Generate JSON data for a 1000 customers to server as documents for search

The Query Context

Two Contexts of Search

Query Context



Filter Context

Two Contexts of Search

Query Context

Filter Context

Query Terms Specification

Search terms as URL query parameters

Search terms within the URL request body

Query Terms Specification

Search terms as URL query parameters

Search terms within the URL request body

Demo

Search using query parameters

Query Terms Specification

Search terms as URL query parameters

Search terms within the URL request body

Demo

Search using the request body to specify parameters

Demo

Source filtering to include only those fields that we're interested in

Demo

Full text queries using:

- match
- match_phrase
- match_phrase_prefix

Relevance in Elasticsearch

The Meaning of Relevance

The search results answered your question or solved your problem

The user understands easily why the search engine retrieved these results

The Meaning of Relevance

Early engines

If the results contain all the search terms then the query was successful

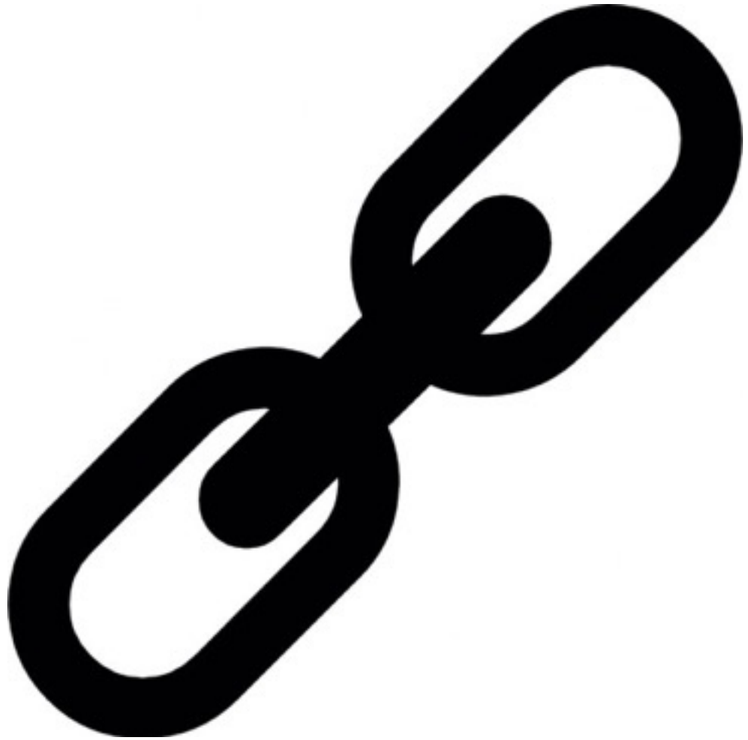
Web search engines

Initial emphasis on high performance, with huge document sets, moved on to finding the one correct document

Second generation engines

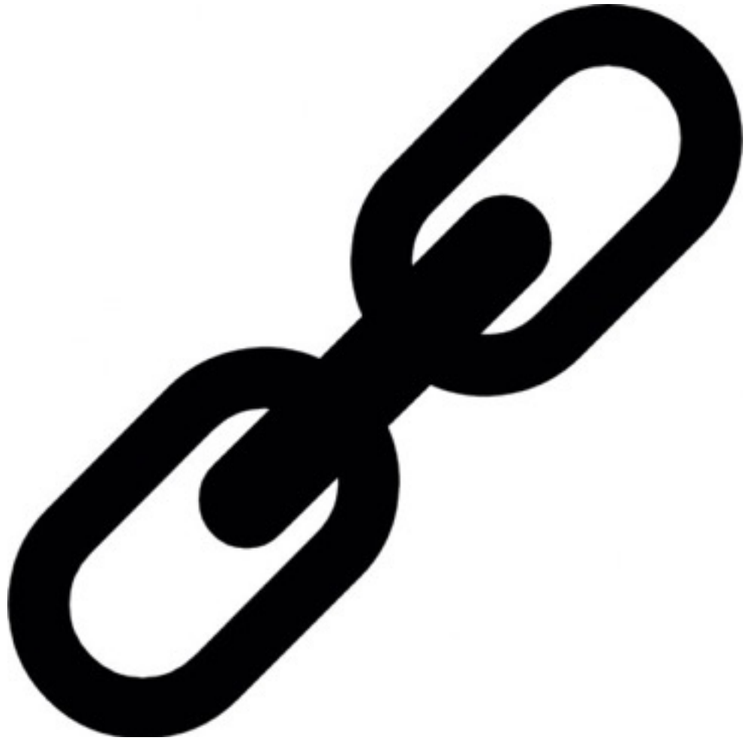
Relevancy score in relation to other documents in the database, better for research queries, rather than lookup

Relevance in Elasticsearch



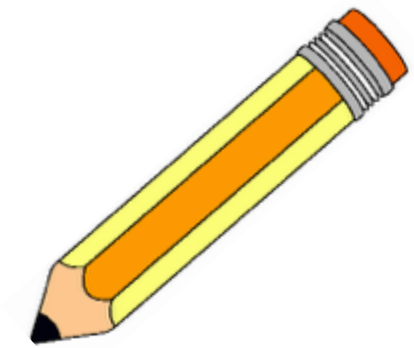
Represented by the _score field in every search result

Relevance in Elasticsearch



Higher the value of _score more relevant the document

Relevance in Elasticsearch

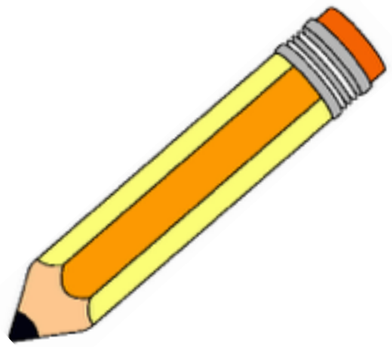


Query clause

Relevance score

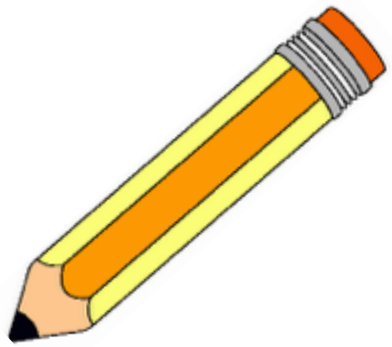
Each document has a different relevance score based on the query clause

Relevance in Elasticsearch



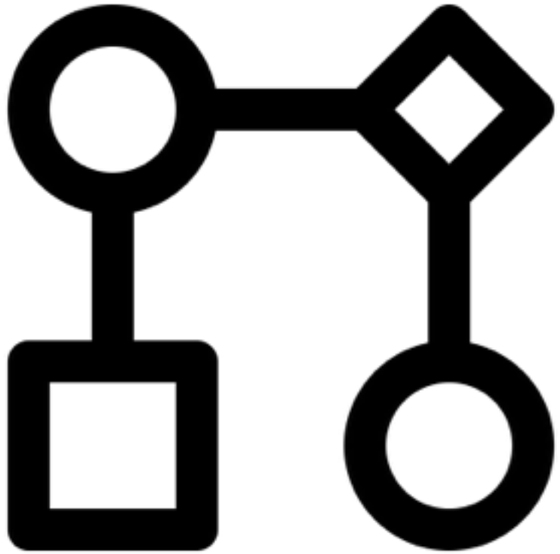
Fuzzy searches might look at how similar the search term is to the word present in the document

Relevance in Elasticsearch



Term searches might look at the percentage of search terms that were found in the document

Elasticsearch Relevance Algorithm



TF/IDF

Term Frequency/Inverse Document Frequency

TF/IDF Relevance Algorithm

Term frequency

How often does the term appear in the field?

Inverse document frequency

How often does the term appear in the index?

Field-length norm

How long is the field which was searched?

Term Frequency

Term frequency

How often does the term appear in the field?

More often, more relevant

A field containing 4 mentions of a term is more relevant than one which has just one mention

Inverse Document Frequency

Inverse document frequency

How often does the term appear in the index?

More often, less relevant

If the term is really common across documents in the index, its relevance for a particular document is low

e.g. stopwords such as “the”, “this”

Field-length Norm

Field-length norm

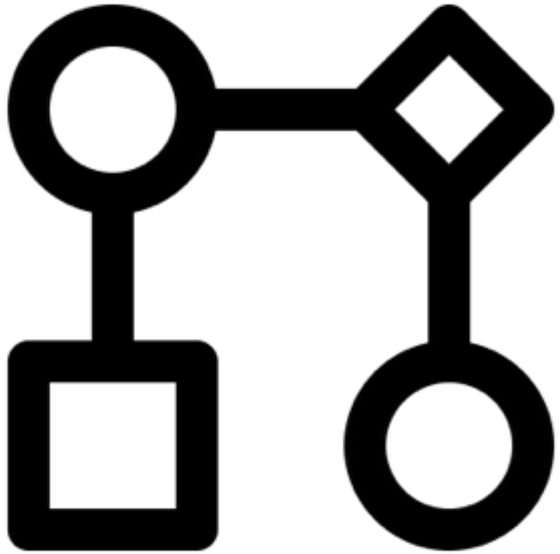
How long is the field which was searched?

Longer fields, less relevant

A term appearing in a longer field is one of a larger set, so less relevant

e.g. words in the title of a book are more relevant than words in the contents

Elasticsearch Relevance Algorithm



The TF/IDF score can be combined with other factors based on the query clause

Relevance in Elasticsearch is calculated using TF/IDF in combination with other factors

The Common Terms Problem

Common Terms

Search for “The quick brown fox”

Common Terms

The word "the" is likely to match a huge number of documents

With low relevance to the actual search

Common Terms

Leaving out stopwords can have unexpected impact

Unable to distinguish between "great" and "not great"

Split Terms: Low and High Frequency

Low: “quick brown fox”

High: “the”

Split Terms: Low and High Frequency

Low: “quick brown fox”

Search for documents which have the rarer terms first

High: “the”

Split Terms: Low and High Frequency

Low: “quick brown fox”

Look for the high frequency terms in the document subset which match the low frequency terms

High: “the”

Split Terms: Low and High Frequency

Improved relevance, good performance

Demo

Common terms queries with
cutoff_frequency specified

Compound Queries: The Boolean Query

Boolean Query

Matches documents by combining multiple queries using boolean operators such as AND, OR

Boolean Query

must

The clause must appear in matching documents

should

The clause may appear in matching documents but mau not sometimes

must_not

The clause must not appear in the document results

filter

The clause must appear in results but results are not scored

Boolean Query

must

The clause must appear in matching documents

should

The clause may appear in matching documents but mau not sometimes

must_not

The clause must not appear in the document results

filter

The clause must appear in results but results are not scored

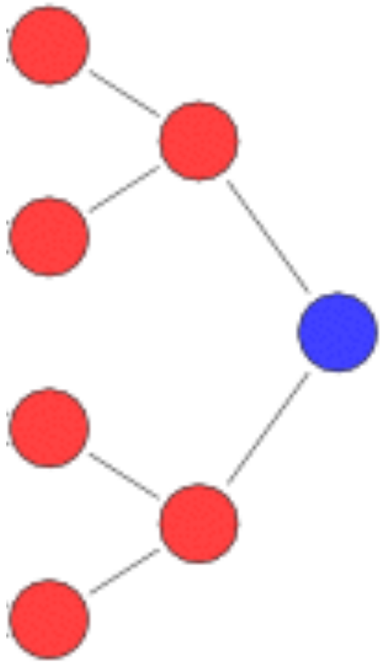
Demo

Run boolean compound queries using:

- must
- should
- must_not

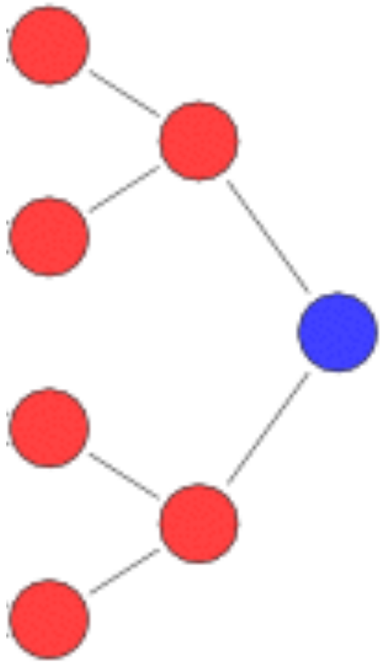
Term Queries

Term Query



The exact term needs to be found in the inverted index for indexed documents

Term Query



The terms found in the index may vary based on how you analyze them

Demo

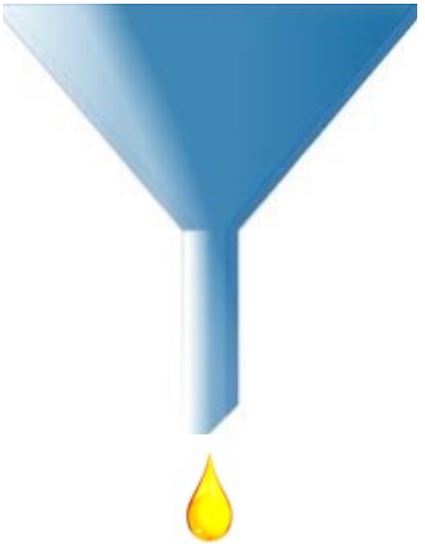
Simple term queries

Boost some terms over others

Filters

Two Contexts of Search

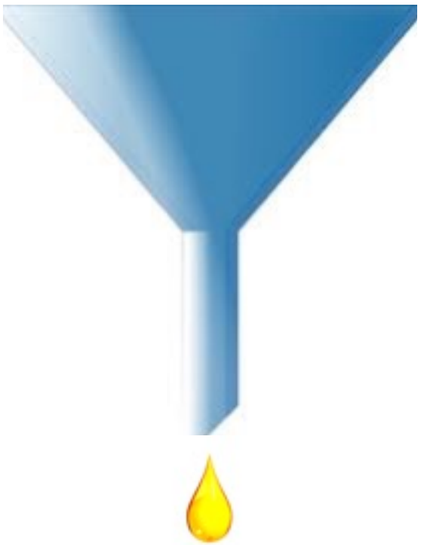
Query Context



Filter Context

Two Contexts of Search

Query Context



Filter Context

Filters

The documents in the result are not scored

Each document responds yes/no to whether it should be included in the result

Demo

Queries to filter documents

Summary

Understood the Query DSL that Elasticsearch uses for search queries

Worked with searches which need relevance and filters where relevance is not required

Worked with full text searches, term searchers, compound searches, filters

Understood the basics of the TF/IDF algorithm for relevance

Implemented all queries using the Elasticsearch REST API