NLP 2

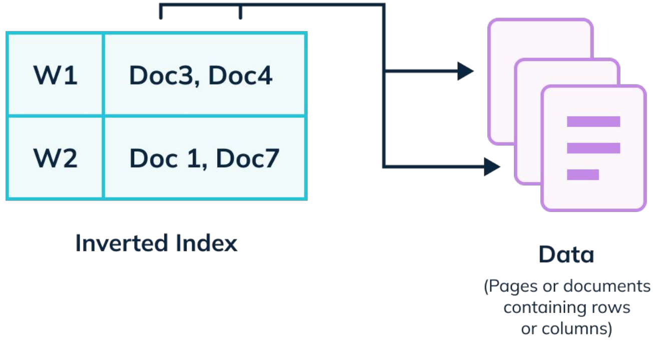
Inteligencia artificial avanzada para la ciencia de datos II Modulo 5 NLP 2

Inverted Index

Inverted Index

An **inverted index** is a data structure used in information retrieval and text search systems to efficiently store and retrieve a large amount of textual data.

It is designed to improve the speed of searching and retrieval operations by mapping terms (words or tokens) to the documents or data records in which they appear.



Software available

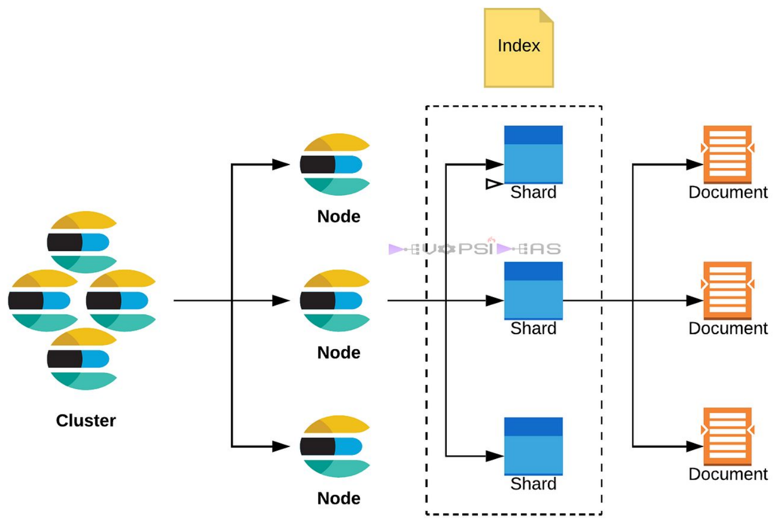
- **Lucene/Solr**: Apache Lucene is a high-performance, full-featured text search engine library written in Java. Apache Solr is an enterprise search platform built on top of Lucene. These tools are widely used for creating search engines and handling complex search operations.
- **Elasticsearch:** Elasticsearch is a distributed, RESTful search and analytics engine built on top of Lucene. It's often used for real-time search and log analytics. Elasticsearch is known for its scalability and ease of use.
- **Opensearch:** Elasticsearch fork supported by AWS, elasticsearch is an open source initiative that didn’t receive earnings for AWS PaaS distribution of Elasticsearch, AWS forked the project and keep it open source to be distributed in the cloud.

ES Architecture

Elasticsearch is a distributed, RESTful search and analytics engine designed for horizontal scalability, real-time search, and large-scale data storage and retrieval.

Its architecture is based on a cluster structure to handle vast amounts of data and to provide high availability and fault tolerance.

Elasticsearch Component Relation



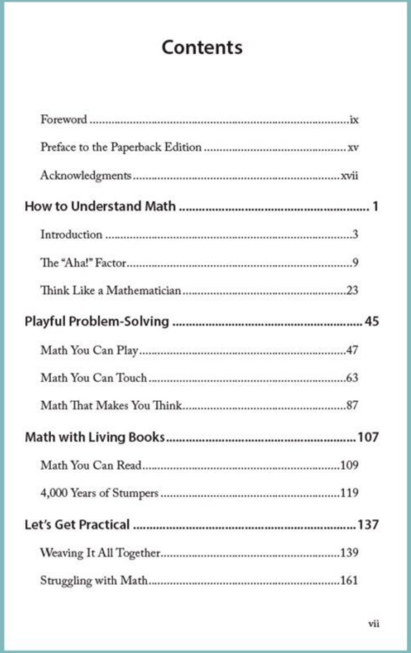
Basics

Imagine you are back in the 80's and you want to search a topic in a book.

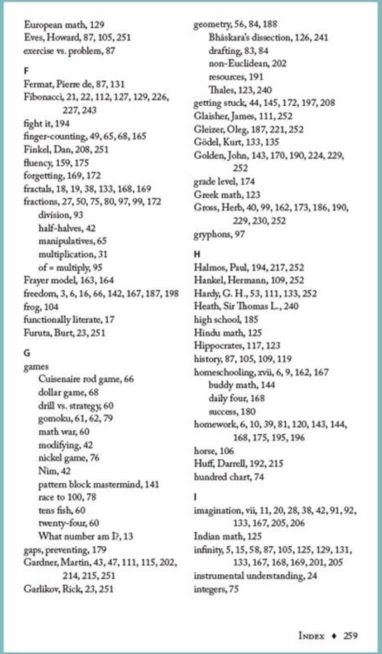
You will go to the index , in order to find the pages in which this topic (Token) appears.

Here are some aspects you would need to know:

1. Someone read the hole book and make a map of the word and the page where it appears.
2. This map is sorted and structured to be efficient to look for



Contents

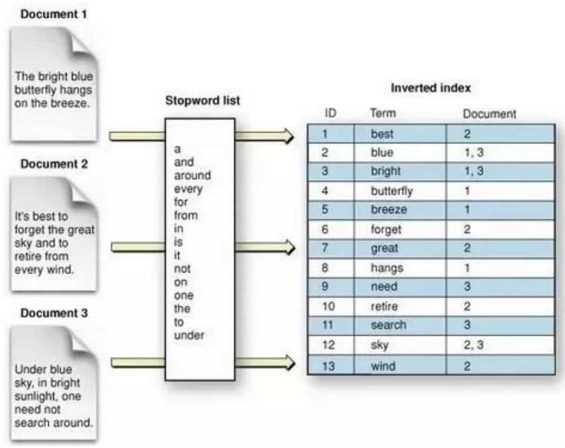


Index time process

Reverse index software is the responsible of building that index for you, so later you can search for a word and retrieve your documents.

This means software has to:

- Process the text and tokenize
- Create the reverse index structure
- Keep some statistics to score the relevance given a sequence.



Document 1

The bright blue butterfly hangs on the breeze.

Document 2

It's best to forget the great sky and to retire from every wind.

Document 3

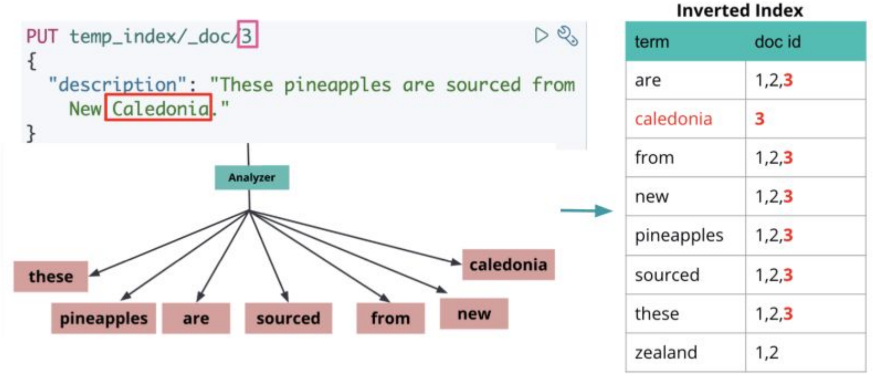
Under blue sky, in bright sunlight, one need not search around.

Stopword list

a
and
around
every
for
from
in
is
it
not
on
one
the
to
under

Inverted index

ID	Term	Document
1	best	2
2	blue	1, 3
3	bright	1, 3
4	butterfly	1
5	breeze	1
6	forget	2
7	great	2
8	hangs	1
9	need	3
10	retire	2
11	search	3
12	sky	2, 3
13	wind	2



```
PUT temp_index/_doc/3
{
  "description": "These pineapples are sourced from
    New Caledonia."
}
```

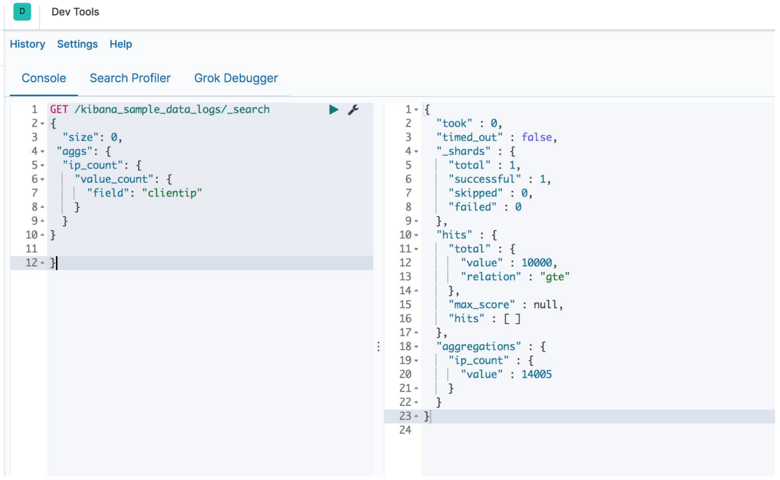
Inverted Index

term	doc id
are	1,2,3
caledonia	3
from	1,2,3
new	1,2,3
pineapples	1,2,3
sourced	1,2,3
these	1,2,3
zealand	1,2

Query time process

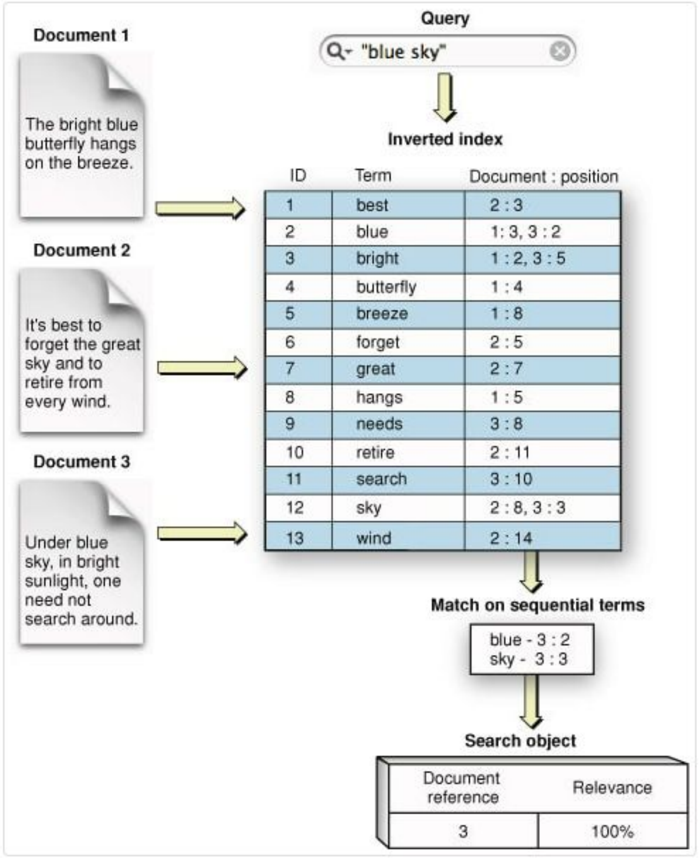
Also, this software will receive the input (sequence) and will be responsible for:

- Tokenize the input
- Compared with the index so it will generate some matches
- Sort results by using a score algorithm (elasticsearch uses TF/IDF as default)



```
GET /kibana_sample_data_logs/_search
{
  "size": 0,
  "aggs": {
    "ip_count": {
      "value_count": {
        "field": "clientip"
      }
    }
  }
}
```

```
{
  "took" : 0,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 10000,
      "relation" : "gte"
    },
    "max_score" : null,
    "hits" : [ ]
  },
  "aggregations" : {
    "ip_count" : {
      "value" : 14005
    }
  }
}
```



Document 1: The bright blue butterfly hangs on the breeze.

Document 2: It's best to forget the great sky and to retire from every wind.

Document 3: Under blue sky, in bright sunlight, one need not search around.

ID	Term	Document : position
1	best	2 : 3
2	blue	1: 3, 3 : 2
3	bright	1 : 2, 3 : 5
4	butterfly	1 : 4
5	breeze	1 : 8
6	forget	2 : 5
7	great	2 : 7
8	hangs	1 : 5
9	needs	3 : 8
10	retire	2 : 11
11	search	3 : 10
12	sky	2 : 8, 3 : 3
13	wind	2 : 14

blue - 3 : 2
sky - 3 : 3

Document reference	Relevance
3	100%

Thanks

Do you have any questions?

emmanuel.paez@tec.mx
Slack #module-5-nlp-1