ASIAN JOURNAL OF PHARMACEUTICAL AND CLINICAL RESEARCH

INNOVARE ACADEMIC SCIENCES
Knowledge to Innovation

Special Issue (April)

Online - 2455-3891
Print - 0974-2441

Full Proceeding Paper

A SURVEY PAPER ON ELASTIC SEARCH SIMILARITY ALGORITHM

NILANJANA DEV NATH[1]*, SHREEKANT JHA[2], JANKI MEENA M[1], SYEDIBRAHIM SP[1]

[1]School of Computing Science and Engineering, VIT University, Chennai Campus, Tamil Nadu, India. [2]Intel Technologies Pvt. Ltd., Bengaluru, Karnataka, India. Email: nilanjana.dvnath@gmail.com

Received: 19 January 2017, Revised and Accepted: 20 February 2017

ABSTRACT

Elastic search is a web search tool in view of Lucene. Apache Lucene is a free and open-source data retrieval programming library. Versatile **s**earch gives a conveyed, multitenant-fit**,** full-content web search tool with a HTTP web interface**,** and pattern free JSON archives. It is created in Java and has been released as open source under the terms of the Apache License. Elastic search can be utilized to pursuit a wide range of records. It gives adaptable hunt, has close continuous pursuit, and backings multitenancy. It is appropriated, which implies that records can be partitioned into shards and every shard can have zero or more duplicates. Every hub has one or more shards and goes about as a facilitator to delegate operations to the right shard(s). Elastic search is like a wrapper on the top of Lucene. In this paper**,** a detailed description of how Lucene's scoring algorithm works and how elastic search uses it as "similarity algorithm**.**"

Keywords: Elastic search, Lucene, Big data, Ranking **a**lgorithm, Indexing, Mapping, Scoring**.**

 DOI: http://dx.doi.org/10.22159/ajpcr.2017.v10s1.19757

INTRODUCTION

The vital of elastic search's savvy web crawler is fundamentally another product extend called Lucene. It is perhaps most straightforward to comprehend elastic search as a part of base constructed adjacent Lucene's Java libraries. In elastic search, everything is identified with the genuine calculations for coordinating content and putting away advanced files of question terms is executed by Lucene. Elastic search itself gives a more practical and conservative API, versatility, and operational apparatuses overhead Lucene's search usage.

Lucene is antiquated in web years, seeing back to 1999. It is additionally to a great degree across the board and built up. Lucene is utilized by unspeakable quantities of organizations, running the degree from enormous partnerships, for example, Twitter, to little new business. Lucene is illustrated, tried, and is broadly considered best of breed in open-source seek programming. The greater part of the balanced exertion clients of elastic search allot to the assignment of inquiry will be identified with utilizing the Lucene APIs elastic search uncovered.

FEW CONCEPTS OF ELASTIC SEARCH

Index

Elastic search stores its information in at least one record. Utilizing likenesses from the SQL world, ordering is like a database. It is utilized to store the records and read them from it. Elastic search utilizes Apache Lucene library to compose also, read the information from the record. Elastic search list might be worked of more than a solitary Apache Lucene index by utilizing "Shards."

Document

Record is the principle element in the elastic search world. At the end, all utilization instances of utilizing elastic search can be conveyed at a point where it is about looking for archives and dissecting them. The record comprises fields, and every field distinguished by its name and can contain one or various values. Every archive may have diverse arrangement of fields; there is no mapping or any structure that is imposed on.

Type

Every archive in elastic search has its sort characterized. This permits us to store different archive sorts in one record, and distinctive mapping for various archives sorts.

Mapping

All reports are dissected before being filed. The information content is separated into tokens, which tokens ought to be sifted out, or what extra handling, for example, evacuating HTML labels, is required. This is the place mapping becomes an integral factor; it holds all the data about the examination chain.

Node

The single example of the elastic search server is known as a NODE. A solitary node in elastic search [3] organization can be adequate for some basic utilize cases. Elastic search is intended to file and hunt our information, so the primary kind of node is the information node. Such nodes hold the information and inquiry on them. The second sort of node is the primary node that functions as chief of the cluster controlling other nodes' work such as indexing. The third node sort is the tribe node, which is new and was presented in elastic search [1]. The tribe node can join numerous clusters and along these lines go about as an extension between them, permitting us to execute all elastic search functionalities on different groups. Much the same as a group, a node is distinguished by a name which as a matter of course is a random universally unique identifier that is doled out to the node at startup. The user can characterize any node name that he needs on the off chance that he has no need for the default. This name is essential for admin purposes where you need to distinguish which nodes in the system compare to which node in elastic search cluster.

Cluster

Cluster is an arrangement of elastic search node that cooperates. The conveyed way of elastic search enables users to effortlessly handle information that is too substantial for a solitary node to handle [5]. A cluster is recognized by a one-of-a-kind name which as a matter of course is "elastic search." This name is critical on the grounds that a node must be a part of a group if the node is setup to join the bunch by its name. Furthermore, it is substantial and superbly fine to have a group with just a single node in it. Besides, user may likewise have numerous independent clusters each with its own special cluster name.

Shards

Clustering permits putting away data volumes that surpass capacities of a single server. To accomplish this necessity, elastic search spreads information to a few physical Lucene lists, these lists are called shards,

and every one of the parts of the record is called sharding [1]. Elastic search can do this naturally, and every one of the parts of the record (shards) is unmistakable to the client as one major record.

Sharding is imperative for two essential reasons:

1. It permits user to a level plane split/scale your content volume.
2. It permits user to disperse and parallelize operations crosswise over shards (possibly on different hubs) in this manner expanding execution/throughput.

The mechanics of how a shard is dispersed, furthermore, how its reports are totaled again into hunt solicitations are totally overseen by elastic search and is straightforward to user as the client.

Replica

Sharding permits pushing more information into elastic search that is workable for a single node to handle. The thought is straightforward to make an extra duplicate of shard, which can be utilized for questions pretty much as unique, essential shard.

Replication is essential for two essential reasons:

1. It gives high accessibility on the off chance that a shard/hub comes up short. Hence, take note of that a copy shard is never dispensed on an indistinguishable hub from the first/essential shard that it was duplicated from.
2. It permits you to scale out your inquiry volume/throughput since hunts can be executed on all copies in parallel.

As a matter of course, every list in elastic search is dispensed 5 essential shards and 1 replica which implies that on the off chance that user has no less than two nodes in his group, his list will have 5 essential shards and another 5 replica shards (1 complete replica) for a sum of 10 shards for each index.

KEY FEATURES OF ELASTIC SEARCH

Elastic search was worked in view of a couple of ideas. The improvement group needed to make it simple to utilize and exceedingly adaptable. These center components are unmistakable in each edge of elastic search. The primary components are as per the following:

1. Reasonable default values that permit clients to begin utilizing elastic search soon after introducing it. This incorporates worked in revelation and auto-design [3].
2. Working in circulated mode as a matter of course. Node accepts that they are or will be a part of the group.
3. Peer to companion engineering without single purpose of disappointment. Nodes naturally associate with different machines in the bunch for the information trade and common checking. This spreads programmed replication of shards.
4. Easily versatile both as far as limit and the sum of information by adding new nodes to the bunch.
5. Elastic pursuit dosage not force limitations on the information association in the record. This permits client to change in accordance with the current information model. Near real time in elasticsearch indicates per-segment search, hence delay between indexing and making it live has reduced drastically. As a result of the circulated way of flexible pursuit, it is difficult to keep away from postponement and brief contrasts between information situated on the diverse hubs. Versatile pursuit tries to diminish these issues, furthermore, give extra components as forming.

BACKGROUND

While elastic search and conventional RDBMSs vary in numerous courses; at the larger amount, a significant number of the center ideas of elastic search have analogs in the RDBMS world (Table 1). All information in elastic search is put away in lists. A list in elastic search resembles a database in a RDBMS: It stores distinctive sorts of records, overhaul them, and look for them. Every report in elastic search is a JSON type, similar to a line in a table in a RDBMS. A record comprises at least zero fields, where every field is either a primitive sort or a more intricate structure. A record has a document sort connected with it; nonetheless, all records in elastic search are without diagram, which implies that two archives of similar sort can have distinctive arrangements of fields. Record sort here is like the RDBMS idea of a table: It characterizes the arrangement of fields that can be determined for a specific document.

Searching in elastic search

Elastic search has its own querying using JSON called Query DSL. This search can be performed in elastic search in two ways: Using a query or using a find of filter. The fundamental distinction between them is that a query ascertains and allots each returned report with the significance score while a filter does not do so. For this reason, seeking by means of filter is speedier than by means of query. The elastic search documentation suggests utilizing questions just as a part of two circumstances: For full-content ventures or when the significance of every outcome in the pursuit is imperative. For straightforwardness, we will utilize term inquiry to depict both filters and queries; be that as it may, our involvement with elastic search is constrained to working as it were with filters; subsequently, we do not report about utilization of queries.

ELASTIC SEARCH WORKFLOW

Elastic search is for some plans and purposes schema-less, which implies that documents can be listed without unequivocally giving a schema. The pattern is a mapping that depicts the fields in the JSON reports alongside their date type and how they ought to be ordered in the Lucene indexes that lie in the engine.

The workflow of how elastic search functions is shown in Fig. 1.

The user uploads files, which are converted to JSON document format. Then, the tokening of words happens and finally the documents are indexed.

Whenever user enters a query text, the first purpose would be to parse the query. Now based on the query passed, the documents are scored. Elastic search similarity algorithm is used to rank the documents based

Table 1: Elastic search versus SQL

Elastic search element	SQL element
Index	Database
Mapping	Schema
Document type	Table
Document	Row

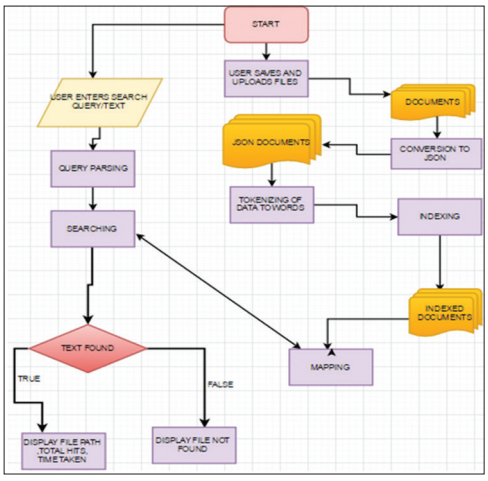

Fig. 1: Workflow of elastic search

on the score. The one having the highest score is passed as the result. If the required document is found, it is successful.

In an elastic search cluster when a request is made, it first passes through a coordinating node. The cluster state has to be known to every node in the cluster. This state contains information about which node consist of which indices and shards. If it is a search request, if can be read either from a primary shard or a replica. The distinct shards present, must contain the search request that has been sent. Following the request, the shards will reflect back the top results obtained and send the consolidated bundle to the coordinator. It is the job of the coordinator to merge the results obtained so that the top global results are obtained and then it is send back to the user.

DIFFERENT INFORMATION RETRIEVAL MODELS USED IN LUCENE

Lucene has proficient and exact search calculations. It recovers the reports questioned in view of their positioning. It gives diverse sorts of questions such as Fuzzy Query, Phrase Query, Boolean Query, Wildcard Query, and Range Query. Lucene uses a combination of the vector space model (VSM) of information retrieval and the Boolean model to determine how relevant a given document is to a user's query [4].

Boolean model

The Boolean model takes into consideration the utilization of operators of Boolean polynomial math, AND, OR, and NOT, for question definition, however, has one noteworthy disadvantage: A Boolean framework is not ready to rank the returned rundown of documents. In the Boolean model, an archive is connected with a set of keywords. Inquiries are likewise articulations of keywords isolated by AND, OR, or NOT/BUT. The retrieval work in this model regards a document as either relevant or not relevant [2].

Vector model

The VSM can best be portrayed by its endeavor to rank records by the similarity between the query and among the documents present. In the VSM, queries and documents present are represented like a vector, and the edge between the two vectors is processed utilizing the cosine similarity. Cosine similarity capacity can be characterized as:

$$\text{Sim}(d_j,q)=\frac{d_j \cdot q}{||d_j||||q||}=\frac{\sum_{i=1}^{N} W_{i,j}\, Wi,q}{\sqrt{\sum_{i=1}^{N} W_{i,j}^2}\sqrt{\sum_{i=1}^{N} W_{i,q}^2}}$$

Where the queries and documents are represented in the form of vectors.

$d_j = (W_{1,j}, W_{2,j}, \ldots, W_{t,j})$
$q = (W_{1,q}, W_{2,q}, \ldots, W_{t,q})$

This model uses the term frequency-inverse document frequency TF-IDF weighting also known as "TF-IDF [2]." These weights will have a TF calculate measuring the recurrence of event of the terms in the archive or query writings and an IDF consider calculating the inverse of the quantity of documents that might contain a document or a query.

SIMILARITY ALGORITHM IN ELASTIC SEARCH

Elastic search uses relevance of a document for determining its search result. Results are generally obtained in decreasing order of relevance of the documents. The relevance is measured using a floating point number called the "score" of the document. This "score" is actually Lucene's practical scoring function. The higher is the score of the document, the more relevance factor it possess.

In general, whenever a query is passed, it will generate a score for each document. It will be different for different types of query clause. The similarity algorithm used in elastic search takes into account the following factors into account:

TF

It determines how often the term appears in the field. The more often it occurs the more relevant it is [3].

IDF

It determines how often the terms appear in the document or in the index. It is "inverse" because the more the terms appear in the index the less relevant it becomes. Hence, the terms whose a number of occurrences are more in a document have lower weight than the terms that occurs less [3].

Field-length norm

It is used to show the length of the field. The longer the length of the field the lesser are the chances of words being more relevant. The term that will appear in a short "title" will carry more weight than the term that occurs in a longer content field.

Query norm

Elastic search uses measures for comparing the combination of query types that is being passed. The factor for measure such combination of query types is called "query normalization factor" or "query norm."

Coord

This gives the measure of matching multiple search terms. If the value of coord is higher, it will in turn increase the overall score of the document.

Boost index and boost query

These boost factors can be used during indexing time and querying time, respectively, to boost up the score. Furthermore, applying a boost on a specific field can have a significant change in score calculation.

Elastic search scoring algorithm is actually a combination of a Boolean model, VSM, and information retrieval model. The documents first pass through the Boolean model and then scoring happens in the VSM.

The scoring formula can be written as:

Score (q,d) = query norm(q) * coord(q,d)*Σ(tf(t in d)*idf(t))^2*

t.getBoost()*norm(t,d)) (t in q)

Where,
q: Query passed
d: Document
Score(q,d): Score of the document with respect to the query
queryNorm(q): Query normalization factor
coord(q,d): Matching terms with respect to the query
tf: Term frequency
idf: Inverse document frequency
get.Boost: Boosting with respect to a particular field.

We actually use boosting for prioritizing our clauses that we pass. Boosting is done by specific factor.

COMPARISON WITH OKAPI BM25

Lucene's practical scoring function internally works as elastic search similarity algorithm. Although it uses scoring function as its standard similarity algorithm, it even has the capability to support other algorithms that are used for scoring. One of such is Okapi BM25.

Okapi BM25 is one the best-known ranking functions which is at par with Lucene's scoring function. It can be compared with Lucene's

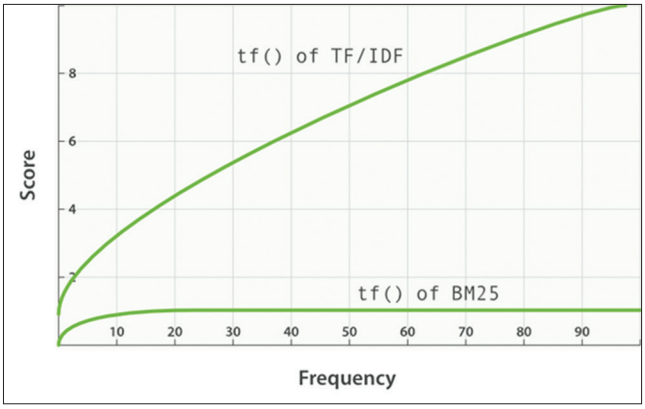

Fig. 2: Term-frequency saturation for BM25

scoring function on the basis of the informational retrieval model that it uses. While Lucene makes the use of the VSM, BM25, on the other hand, uses probabilistic relevance model [6].

The standard analyzer of elastic search does not remove the stop words and hence this creates an artificial boost in their weights which is irrelevant. However, using BM25, a certain upper limit is set which makes the distinction of stop words and the terms that occur lesser number of times in document to have the same impact. This is known as non-linear TF saturation.

Fig. 2 shows that terms that show up 20 times in an archive have practically an indistinguishable effect from terms that show up a thousand times or more.

Another striking difference between the standard Lucene score in elastic search is that the longer fields have practically lesser weight than the shorter ones which usually have more weight. In other words, the recurrence of a term in a field is counterbalanced by the length of the field. BM25 takes shorter fields to have more weight than the loner ones, but it takes into account the average length of each field separately. It can easily compare a title field that is short and a title field that is long.

Apart from all these factors, BM25 also has some tuning factors which are placed in such a manner that it is suitable for the collection of documents present.

CONCLUSION AND FUTURE WORK

Elastic search is actually powered by Lucene. It is analogous to a car and its engine. We see that the concept of inverted indexing in elastic search makes searching easier and less complicated. Its schema-free architecture helps in detecting the data structure easily and thus becomes more searchable.

Elastic search can also be integrated with BM25, which is similar to the default similarity except some of the differences such as the shorter documents are more heavily weighted. The TF saturation makes more accurate in ranking the documents based on the score. Elastic search can been integrated with any of the big data platforms such as Hadoop [7] and large data can undergo heavy indexing operations. Lucene offers phenomenal genuine usefulness such as hit highlighting, spell checking, tokenizing, and breaking down; however, one of the intense and oft-utilized components is boosting. Now-a-days the websites provide some level of search usefulness to them which go from seeking plain content substance inside the site to particular substance covered up inside documents.

REFERENCES

1. Kononenko O. Mining Modern Repositories with Elasticsearch. Hyderabad, India: MSR; 2014.
2. Sematext. Elasticsearch Refresh Interval vs. Indexing Performance. Available from: http://www.bit.ly/1iZoPGc.
3. Divya MS, Goyal SK. Elastic search: An advanced and quick search technique to handle voluminous data. COMPUSOFT Int J Adv Comput Technol 2013;2(6):171-5.
4. Long B, Chapelle JB, Zhang Y. Ranking through expected loss optimization. IEEE Trans Knowl Data Eng 2015;27(5):267-74.
5. Lin J, Ryaboy D, Weil K. Full-text indexing for optimizing selection operations in large-scale data analytics. San Jose, California, New York, USA: ACM; 2011.Available from: http://www.acm.org/publications.
6. Unpluggable Similarity. Available from: https://www.elastic.co/guide/en/elasticsearch/guide/current/pluggablesimilarites.htm.
7. Butler MH, Rutherford J. Distributed Lucene: A Distributed Free Text Index for Hadoop. HP Laboratories, HPL; 2008.