Gümüşhane University Journal of Science

GUFBD / *GUJS* (2023) 13(2): 386-404 doi: 10.17714/gumusfenbil.1213317 Research Article

A comparison of Apache Solr and Elasticsearch technologies in support of large-scale data analysis

Büyük ölçekli veri analizini desteklemek için Apache Solr ve Elasticsearch teknolojilerinin karşılaştırması

Ayşenur DENİZ, Muhammed Mehdi ELÖMER, Ahmet Arif AYDIN*[*]

Inonu University, Faculty of Engineering, Department of Computer Engineering, 44280, Malatya

• Received: 01.12.2022 • Accepted: 10.03.2023

Abstract

In the era of big data, data has never been more important because it contains hidden insights. Additionally, it is necessary and challenging to extract usable information from enormous volumes of data. When attempting to perform data processing and analytics in a variety of domains, developers of data-intensive systems have consequently met several challenges. In addition, full-text search is one of the most significant components of big data processing and analytics for discovering fragments of required data among large volumes of data. Due to the importance of the subject, this article begins with an examination of the characteristics, capabilities, and technical comparisons of full-text search technologies, followed by a systematic comparison of Apache Solr and Elasticsearch in terms of indexing times and queries on three separate datasets. According to our findings, based on default configuration, Apache Solr has better performance when looking at indexing times measured on three machines with different hardware specifications. Likewise, Apache Solr outperforms Elasticsearch in seven out of ten search queries. Regarding our results, on computers with restricted hardware resources, we recommend utilizing Apache Solr instead of Elasticsearch. In addition, this study provides researchers and developers of data-intensive systems with a complete comparison and suggestions for choosing the most effective full-text search engine for their task.

Keywords: Apache Lucene, Apache Solr, Big data, Elasticsearch, Full-text search, Searching

Öz

Büyük veri çağında, gizli içgörüler içerdiği için veriler hiç bu kadar önemli olmamıştı. Ayrıca, çok büyük hacimli verilerden kullanılabilir bilgileri çıkarmak zaruri ve zordur. Çeşitli alanlarda veri işleme ve analitiği gerçekleştirmeye çalışırken, veri yoğunluklu sistem geliştiricileri çok çeşitli zorluklarla karşılaşmaktadır. Ayrıca, tam metin arama, büyük veriler içinde gerekli verilerin istenilen kısımlarını ortaya çıkarmak için büyük veri işleme ve analitiğinin en önemli bileşenlerinden biridir. Konunun önemi nedeniyle bu makale, tam metin arama teknolojilerinin özelliklerinin, yeteneklerinin ve teknik karşılaştırmalarının incelenmesiyle başlamakta, ardından Apache Solr ve Elasticsearch'ün indeksleme süreleri üç ayrı veri seti kullanılarak sorgulama açısından sistematik bir karşılaştırması ile devam etmektedir. Bulgularımıza göre, karşılaştırılan teknolojilerin varsayılan konfigürasyonlarını baz alarak, Apache Solr, farklı donanım özelliklerine sahip üç makinede ölçülen indeksleme sürelerine bakıldığında daha iyi performansa sahiptir. Aynı şekilde, on arama sorgusunun yedisinde Apache Solr Elasticsearch'ten daha iyi performans göstermektedir. Sonuçlarımıza göre, kısıtlı donanım kaynaklarına sahip bilgisayarlarda, Elasticsearch yerine Apache Solr kullanmanızı öneririz. Buna ek olarak, bu çalışma, araştırmacılara, veri yoğunluklu sistem geliştiricilerine, gerçekleştirecekleri görevleri için en uygun tam metin arama teknolojisini seçmeleri için eksiksiz bir karşılaştırma ve öneriler sağlamaktadır.

***Anahtar kelimeler**: Apache Lucene, Apache Solr, Büyük veri, Elasticsearch, Tam metin arama, Arama*

[*] Ahmet Arif AYDIN; arif.aydin@inonu.edu.tr

1. Introduction

With the advancement of technology, ubiquitous internet access, and affordability, the rate of data growth continues to accelerate. Various sources, services, software tools, and hardware devices have been generating large amounts of data in various formats, sizes, and speeds. The data that reached exabytes and zettabytes became known as "Big Data" (Halevi & Moed, 2012). Big data is defined as large amounts of data that cannot be processed at once and require sophisticated processing tools, technologies, and methods. The daily production of vast quantities of diverse data types from numerous sources (Domo Company, 2022) increases the significance of big data since data includes hidden insight for organizations to stay competitive in the job market. In the context of big data, big data is described with five Vs. These properties consist of *volume* (amounts of data), *velocity* (speed of incoming data), *veracity* (trustworthiness), *variety* (various types and different forms), and *value* (beneficial information) (Lashkaripour, 2020). Each one of these characteristics has introduced various challenges for the developers of data-intensive systems. To handle these challenges, various tools have been developed to handle streaming, storage, and analytics with the purpose of fulfilling domain requirements and user needs (Rao et al., 2018).

In this era of big data, gleaning useful information out of large amounts of data in a reasonable amount of time is crucial since time is money. Moreover, one size does not fit all; various demands, different data types, diverse time restrictions and priorities, and the available resources of the underlying hardware all have an impact on software developers' choices on technology, methods, and approach selection when developing a data-intensive system to perform actions quickly as demanded by the user. Thus, choosing a set of appropriate tools for a data processing job is a vital task since the technology choice impacts the entire data processing task. In addition, searching is another important concept to find related portions of a data set regarding user requests (Barrenechea et al., 2017). On the other hand, it is very difficult to get results quickly with traditional methods due to the large amount of data. Therefore, new technologies are needed to perform analysis on large amounts of big data.

Searching is one of the most important concepts in the context of big data research, and it is performed by the built-in searching capabilities of various data processing tools, ad-hoc codes written in various programming languages, or full-text search technologies such as Apache Lucene, Apache Solr, and Elasticsearch. Full-text search technologies are crucial and have been implemented in a variety of data-intensive systems (Anderson et al., 2015), information retrieval applications (Wang et al., 2022), including search engines, e-commerce applications, education platforms (Y. Aldailamy et al., 2018), smart city and IoT applications (Bellini et al., 2019), including search engines, e-commerce applications, social networking platforms, mobile banking apps, and video streaming services. Due to the significance of full-text search and utilizing the appropriate technology for searching activities, the purpose of this article is to compare Apache Solr and Elasticsearch technologies. We examined the indexing and search times of both technologies. We have calculated indexing times for each technology using three distinct datasets and three distinct machines. Moreover, search times have been computed using 10 queries for each technology on one machine which has good indexing performance.

The organization of the paper is as follows: In Section 2, summaries of relevant research are provided. The comparison and overview of Full-Text Search Technologies, Apache Solr, and Elasticsearch is presented in Section 3. Section 4 provides details on the environments utilized, datasets, and queries. Section 5 presents Apache Solr and Elasticsearch's indexing and search results. In comparison to previous research, Section 6 evaluates the indexing and search performance of Apache Solr versus Elasticsearch. In Section 7, a conclusion is presented by summarizing the paper's contributions.

2. Related works

There are several research applications that employ Apache Solr and Elasticsearch to perform a variety of tasks in big data processing and analytics systems. This section particularly provides articles that compare the technologies Apache Solr and Elasticsearch.

In Oussous and Benjelloun (2022), the authors provide a detailed analysis of full-text search. It provides a comprehensive comparison of search engines, particularly Solr and Elasticsearch, based on relevant publications in the relevant literature. The authors analyzed existing search and indexing technologies using a variety of factors, including use cases, indexing performance, searching, sharding, and rebalancing, data visualization, and data sources, among others. In this work, the authors examine previous research on full-text search and present a comprehensive technical comparison.

In Elasticsearch vs. Solr Performance: Round 2 (2015), querying and indexing speeds, ease of use, and difficulties between Solr and Elasticsearch are analyzed in terms of configuration forms and architectures. Similar to our study, indexing and search speed were compared for both technologies. Although similar ideas can be reached at some points, the testing environment is different from our working environment.

In Luburić and Ivanovic (2016), the authors examine the common features and differences between Apache Solr and Elasticsearch by comparing them. Luburić and colleagues presented a detailed and comprehensive review by examining other published studies based on the comparison of these two technologies. Similarly, in our study, we presented a comparison of Apache Solr and Elasticsearch based on their capabilities for indexing performance on three machines (see Table 2 for hardware configurations) and we have conducted the search queries through one machine that performed better indexing performance with respect to other two machines.

In Kılıç and Karabey (2016), Apache Lucene, Apache Solr, and Elasticsearch are mentioned based on their working principles. Then, Solr and Elasticsearch are examined independently with regard to capabilities such as full-text searching, advanced filtering, Rest API, and content insertion. In other words, Solr and Elasticsearch are compared in terms of effectiveness, usability, speed, and security, and the advantages and disadvantages of both search engines are outlined. Finally, both technologies are examined in terms of security configurations and controls. A similar aspect of our study is that it compares both technologies in technical terms. In contrast to this article, our study does not compare technology in terms of security.

In Hansen et al. (2018), the memory and time consumption, functionality, and indexing efficiency of the full-text search processes of the search engines Solr and Elasticsearch are compared. By analyzing the outcomes of a series of experiments, the authors demonstrate that Elasticsearch is superior over Solr in terms of index size and indexing time, whereas Solr performs better with large-scale datasets. This paper is the most similar to our study in terms of methodology and testing among the related literature. In contrast to other test environments, ours is based on the most recent versions and capabilities of Apache Solr and Elasticsearch technologies.

In Voit et al. (2017), full-text search technologies, including Apache Solr, Elasticsearch, Sphinx, and Xapian, are compared in terms of indexing, searching, and several technical characteristics. However, it is not an article in which a comparison is made and detailed. The conference work D.S. (2016) mentioned by the authors has been written in Russian. Due to this, it has not been examined in depth.

In Yurtsever et al. (2022), the authors developed an application on image texts in Big Data. The best fit-for-purpose technology is sought for a fast and effective search in these image texts. Therefore, a comparison is made between Apache Solr and Elasticsearch technologies. In a similar way to our study, the authors compared the search times of the two technologies in this study.

In Gonçalves and Sunye (2020), the authors provide a benchmark for Apache Solr and Elasticsearch using the DSpace repository platform. They compare the advantages and disadvantages of these two technologies in terms of indexing time, size of RAM used, and index size created. Unlike our work, median precision-at-10 and binary preference metrics are also compared for all search queries in Apache Solr or Elasticsearch.

3. Background: technologies for full-text search

This section provides a background on full-text search and related popular technologies. Full-text search is a widely used search method for massive datasets. It appears in numerous areas, such as web search engines, corporate search sites, and various data-intensive systems. In full-text search, there are two major steps: indexing and searching. Initially, a dataset is indexed, and then based on the created indexes, various searching requests are performed. Full-text search applications mostly use inverted indexes. Moreover, Figure 1 shows the popularity of the top four technologies (Apache Solr, Elasticsearch, OpenSearch, and Splunk) in the rankings for search engines by DB-Engines (DB-Engines, 2022), and Figure 1 is created with data provided by Google Trends (Google Trends, 2022). All four technologies are based on the full-text search method. Elasticsearch is at the top of the rankings, while Apache Solr is the third most used search engine.

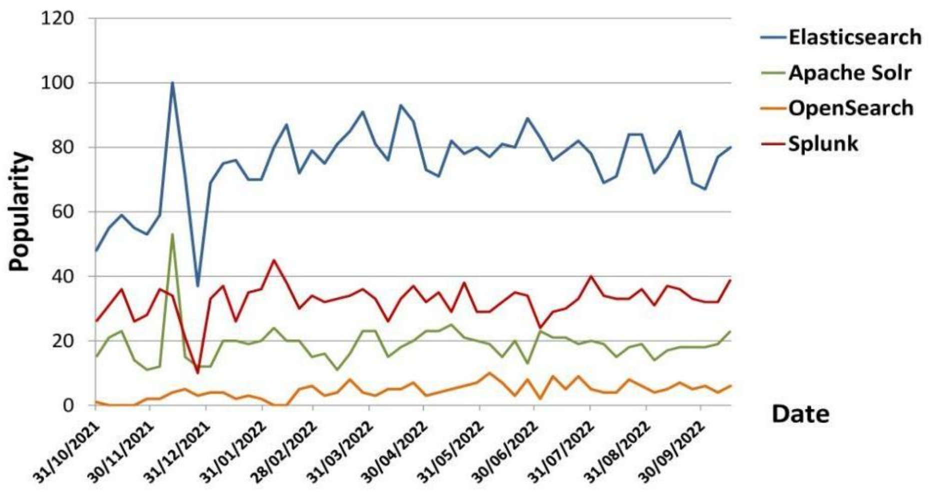


Figure 1. Popularity of full-text search tools over the past year (Google Trends, 2022)

The following subsections provide detailed information about Apache Lucene, Apache Solr, and Elasticsearch technologies.

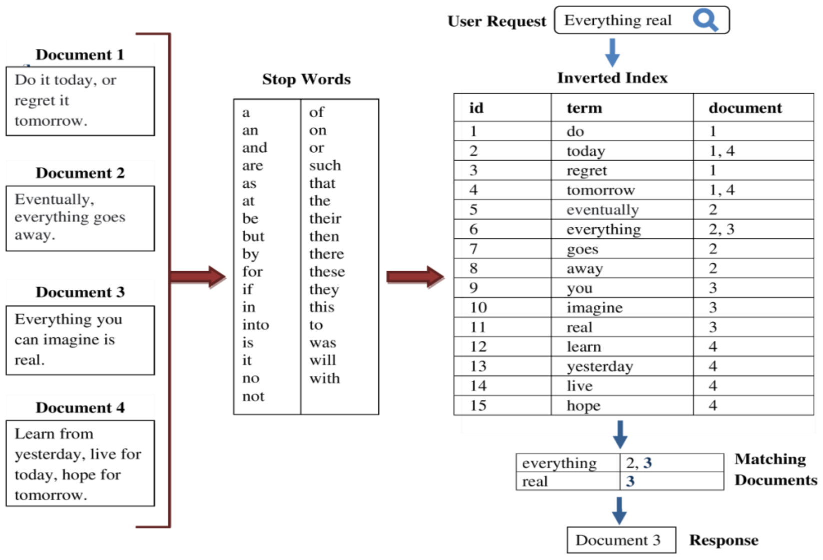
3.1. Apache Lucene

Apache Lucene is a Java-based, open-source search library endorsed by the Apache Software Foundation (Apache Lucene, 2022). It provides robust indexing and search functionality. It is therefore a prominent library in terms of full-text search. Additionally, Apache Lucene has been implemented in a wide range of programming languages, including Python (PyLucene), .Net (DotLucene), and C (CLucene). Furthermore, there are other studies including apps like PyLucene (Lokoč et al., 2021) and DotLucene (Lakhara & Mishra, 2017).

Apache Lucene is essential since it forms the basis for the Apache Solr and Elasticsearch search engines. In addition, Java developers may utilize Apache Lucene with ease. Also, the helpful and user-friendly interfaces that Apache Solr and Elasticsearch have created may be viewed as abstractions that facilitate the direct usage of Apache Lucene. During the development of Apache Lucene, system requirements are considered based on the number of documents, the number of hits, the size of the documents, and so on. It can provide scalable and efficient indexing. For instance, it utilizes less RAM with only 1 MB of heap, and the index is about one-third the size of the indexed data. Moreover, the indexing process is considerably optimized.

Apache Lucene provides an inverted index mechanism that enables fast and effective access to search engines. In Figure 2, an example of the inverted index mechanism is shown. Documents provide the set of data to be indexed. In other words, the documents can be thought of as each row in a relational database. The expressions in the stop word set are ignored, and the associated documents for each term are added to the inverted index structure. For example, as in Figure 2, the expression “everything real” is searched. For this purpose, matching documents are found for every word searched in the inverted index. In these found documents, the document common to each expression is filtered, and a response is returned. Consequently, document 3 was returned as the appropriate answer.

Examining the Apache Lucene library's search capabilities reveals why it is the foundation of two popular technologies, such as Apache Solr and Elasticsearch. Apache Lucene provides an accurate, powerful, and efficient search, and one of the prominent features (Apache Lucene, 2022) are providing ranked searching on the principle that best results are returned first; supporting many query types, such as range queries, phrase queries, and proximity queries; capability of searching through any field such as title, name, age; sorting capability by any field and enabling flexible faceting, highlighting, joins and result grouping.


User Request
Everything real
Document 1
Do it today, or regret it tomorrow.
Document 2
Eventually, everything goes away.
Document 3
Everything you can imagine is real.
Document 4
Learn from yesterday, live for today, hope for tomorrow.
Stop Words
a an and are as at be but by for if in into is it no not
of on or such that the their then there these they this to was will with
Inverted Index
Matching Documents
Response
Document 3

id	term	document
1	do	1
2	today	1, 4
3	regret	1
4	tomorrow	1, 4
5	eventually	2
6	everything	2, 3
7	goes	2
8	away	2
9	you	3
10	imagine	3
11	real	3
12	learn	4
13	yesterday	4
14	live	4
15	hope	4

everything	2, **3**
real	**3**

Figure 2. Mechanism of the inverted index

Figure 3 demonstrates the relationship between Apache Lucene and the two tools, Apache Solr and Elasticsearch. Prior to indexing a dataset using the aforementioned methods, the dataset is pre-processed by removing noise or filling in missing values, depending on the goals. Each technique then employs Apache Lucene to index datasets. Lastly, different search requests from users are answered by using the search APIs of both technologies on Apache Lucene indexed and stored datasets.

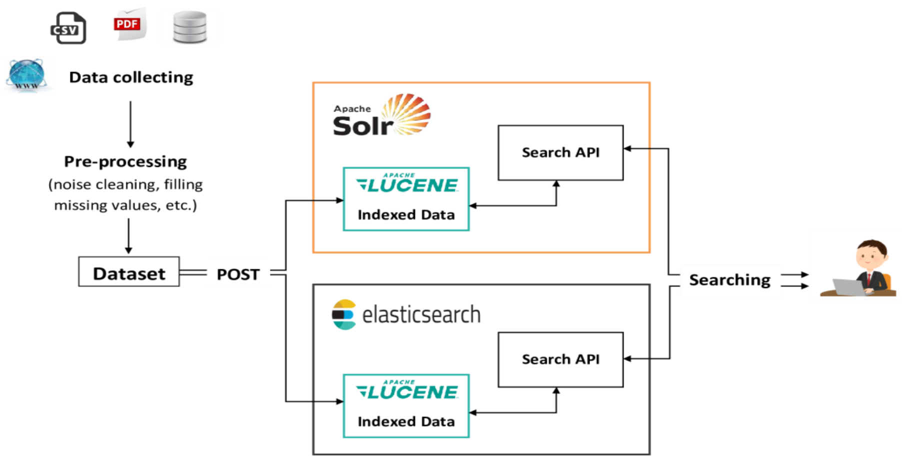


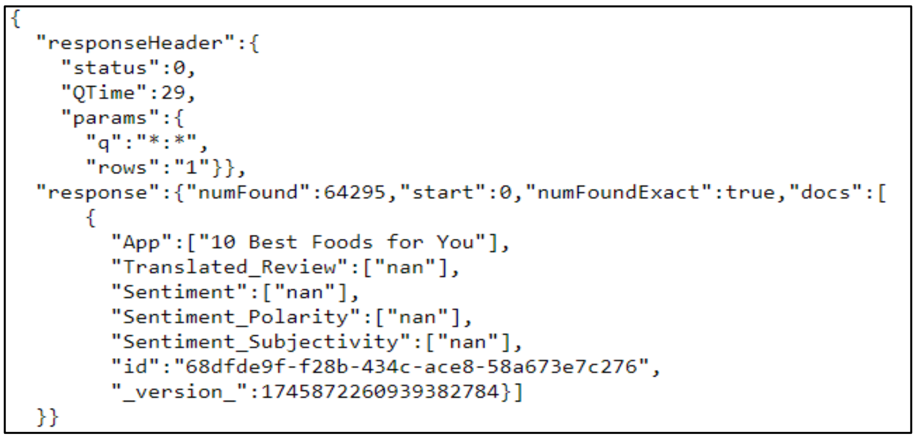
Figure 3. Apache Lucene and related technologies

3.2. Apache Solr

Apache Solr is an open-source full-text search engine that is built on Apache Lucene. It is designed to perform a high-performance search of large datasets. One of its major advantages is that it provides a convenient interface. Apache Solr works on text-based structured data. The data is basically indexed as JSON, but it can also be used for other formats such as CSV and PDF. Apache Solr provides three options for indexing datasets: through the Solr dashboard, command-line curl, and a client API.

Apache Solr is able to respond rapidly to complex queries with multi-domain and faceted searching. It also has powerful mathematical expressions that collapse and aggregate the results. Undoubtedly, its most important characteristic is that it can be used effectively in large-volume and data-intensive applications. It provides the ability to work in a distributed system across multiple servers when running only one server is a problem on large datasets. Nowadays, some well-known sites that use Apache Solr, such as Macy's, eBay, and Zappo's, are examples of its use in high-volume and data-intensive applications (Resources Apache Solr, 2022).

Figure 4 provides an example response that is in a key-value JSON format and includes the following responseHeader, and response. The responseHeader key reports the query parameters (params), the processing time performed (QTime), and whether the operation was carried out without error (status). Additionally, the response key provides information about the number (numFound) and content (docs) of documents found following the request.



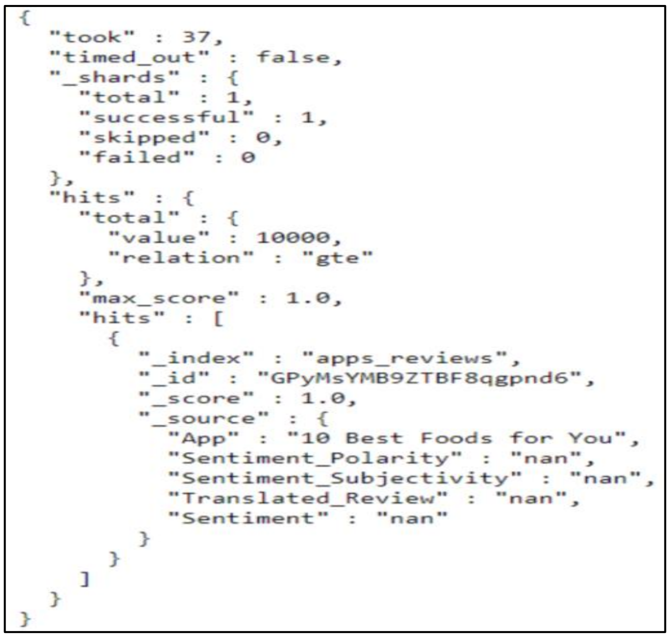
```
{
  "responseHeader":{
    "status":0,
    "QTime":29,
    "params":{
      "q":"*:*",
      "rows":"1"}},
  "response":{"numFound":64295,"start":0,"numFoundExact":true,"docs":[
      {
        "App":["10 Best Foods for You"],
        "Translated_Review":["nan"],
        "Sentiment":["nan"],
        "Sentiment_Polarity":["nan"],
        "Sentiment_Subjectivity":["nan"],
        "id":"68dfde9f-f28b-434c-ace8-58a673e7c276",
        "_version_":1745872260939382784}]
  }}
```

Figure 4. An example of Apache Solr response

3.3. Elasticsearch

Elasticsearch is a free and open-source JSON-based search and analytics engine written in Java and based on Apache Lucene. Elasticsearch is an effective tool for storing, searching, and analyzing textual, quantitative, geographic, structured, and unstructured data. Elasticsearch, recognized for its extensive and powerful REST APIs, distributed architecture, real-time support, efficiency, flexibility, and scalability, supports 34 languages. Using Elasticsearch, Logstash, and Kibana, it facilitates search, analysis, and visualization.

In addition, Elasticsearch is well-suited for time-sensitive use cases like security analytics and infrastructure monitoring (Elastic Installation and Upgrade Guide [8.4], 2022). Today, it is utilized in projects such as Elasticsearch, Mozilla, Foursquare, and GitHub for content search, data analysis, and queries. Moreover, Figure 5 shows an example of an Elasticsearch response, which contains the time the request was submitted (took), the status of the timeout (timed out), and the quantity and content (hits) of documents discovered.



```
{
  "took" : 37,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 10000,
      "relation" : "gte"
    },
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "apps_reviews",
        "_id" : "GPyMsYMB9ZTBF8qgpnd6",
        "_score" : 1.0,
        "_source" : {
          "App" : "10 Best Foods for You",
          "Sentiment_Polarity" : "nan",
          "Sentiment_Subjectivity" : "nan",
          "Translated_Review" : "nan",
          "Sentiment" : "nan"
        }
      }
    ]
  }
}
```

Figure 5. An example of Elasticsearch response

3.4. Feature Comparison

This section describes the characteristics of Apache Solr and Elasticsearch. The following terminology is utilized by both systems: field, value, document, node, core/index, collection, and documents/hits. The "field" describes how the data is defined, whereas the value provides the information that corresponds to this definition. It is also possible to describe it as a field-value pair. A "core" represents an Apache Solr logical index. In other words, an instance of Solr. And this term's equivalent in Elasticsearch is "indice". A document with one or more fields is considered an index unit. A "collection" consists of one or more documents, and each collection has shards or cores that create a single logical index. Additionally, each collection can provide different and flexible settings and schema designs. A "node" describes a single instance of Apache Solr or Elasticsearch that is operating on a physical system or server. Multiple nodes comprise a "cluster", with each node containing an Apache Solr or Elasticsearch instance. In Apache Solr, the result of a query is referred to as "documents", but in Elasticsearch it is referred to as "hits".

Table 1 gives a comparison of Apache Solr and Elasticsearch's general features. The comparison is based on the following characteristics: first release date, built-in functionality, developer, current release, access protocols, supported data formats, and client libraries. Elasticsearch supports a greater variety of programming languages than Solr. Versions 9.0.0 of Apache Solr and 8.4.2 of Elasticsearch were utilized. The Apache Solr 9.X version used requires a minimum of Java 11 and the Elasticsearch version requires a minimum of Java 17. In our experiments, we utilized Java 17.

Table 1. Feature comparison of Apache Solr and Elasticsearch

Feature	Apache Solr	Elasticsearch
Release year	2004	2010
Built on	Apache Lucene	Apache Lucene
Developer	Apache software foundation	Elastic
License	Open source	Open source or commercial
Current version	9.0.0	8.4.2
Web admin interface	Built-in	With apps (Kibana, Marvel, etc.)
Access protocols	REST API (used HTTP)	REST API (used HTTP)
Data importing tools	Data import handler (DIH), Apache Tika (PDF, Word, etc.)	Kibana (JSON, CSV, NDJSON)
Supported data formats	CSV, XML, JSON	JSON
Client libraries	Java, Python, Ruby, PHP, C# / .NET, Scala, Perl, JavaScript / JSON, Node.js, Clojure, Go, Rust, R, C++, Lua	B4J, C++, Clojure, ColdFusion (CFML), Erlang, Go, Haskell, Java, JavaScript, Kotlin, Lua, .NET, Perl, PHP, Python, R, Ruby, Rust, Scala, Smalltalk, Swift, Vert.x
Operating system compatibility	All OS includes Java VM	All OS with includes VM

Elasticsearch has three configuration files that are elasticsearch.yml (editing properties), jvm.options (configuring JVM settings), and log4j2.properties (configuring logging). In the same way, there are basically three configuration files in the Solr Core that are solrconfig.xml (configures high-level behaviors), schema.xml/managed-schema.xml (arrangement of various definitions), and core.properties (defines certain characteristics). When these files are examined, the default values and definitions of some properties for a single node can be explained as follows:

- Apache Solr searches for 10 results by default and prints them to the screen.
- Elasticsearch searches for up to 10,000 but only prints 10 data to the window.
- The RAM buffer size in Apache Solr is 100 MB and the maximum number of documents that can be buffered is 1000.

- The minimum buffer size for Elasticsearch is 48 MB.
- The cache size is defined as unlimited in the default settings of Elasticsearch.
- In the default settings of Elasticsearch, the cache size is defined as unlimited. For Apache Solr, it is 512 MB as an initial value.

In addition, Apache Solr's security settings are initially disabled, but Elasticsearch's default is enabled. Therefore, the *xpack.security.enabled*, *xpack.security.enrollment.enabled*, and *xpack.security.http.ssl* properties in elasticsearch.yml have been updated to false. Thus, the security settings for both technologies were disabled. In this study, with the purpose of performing a fair comparison, we utilized the default configuration (except security settings) for both technologies.

4. Environment, datasets and queries

This section includes details about our environment setup, settings, datasets, and queries. Each one is explained in detail next.

4.1. Hardware specification and configuration

The indexing and searching operations were performed on three computers with distinct RAM, processors, hard drives, CPU cores, and operating systems. The specifications of the machines utilized in the experiments are presented in Table 2.

Table 2. Features of the three machines

Component	Machine 1	Machine 2	Machine 3
Memory (RAM)	32 GB	16 GB	12 GB
Processor (CPU)	Intel(R) Core(TM) i7-12700H 2.50 GHz	Intel(R) Core(TM) i7-9750H 2.60 GHz	Intel(R) Core(TM) i5-4210M 2.60 GHz
Storage	1 TB M.2 3.0 SSD	256 GB M.2 3.0 SSD	256 GB SATA 3.0 SSD
CPU core	14 cores	6 cores	2 cores
Operating system	Win 11	Win 11	Win 10

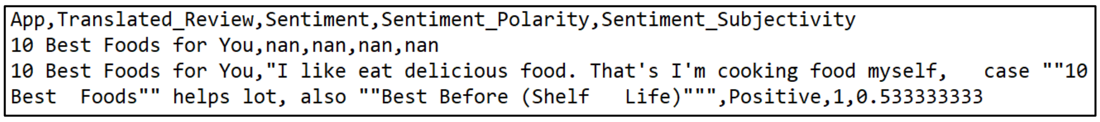
4.2. Datasets

In this section, we present the three datasets utilized to perform a fair comparison between Apache Solr and Elasticsearch technologies. Also, detailed information about the dimensions and contents of the three datasets are presented next and Table 3 provides an overview of the datasets.

Table 3. Features of the three datasets

	Dataset 1	Dataset 2	Dataset 3
Name	Google Play Store Apps (Google Play Store Apps, 2022)	Web of Science (Kowsari et al., 2018)	Dota 2 Matches (Dota 2 Matches, 2022)
Size in MBs	~ 17 MB	~ 75 MB	~ 300 MB
Cardinality	64295	46985	1500000
Data types	Text and numeric	Text	Numeric
Fields	App, Translated_Review, Sentiment, Sentiment_Polarity, Sentiment_Subjectivity	Y, Y2, Y1, Domain, Area, Keywords, Abstract	match_id, player_slot, buybacks, damage, deaths, gold_delta, xp_end, xp_start

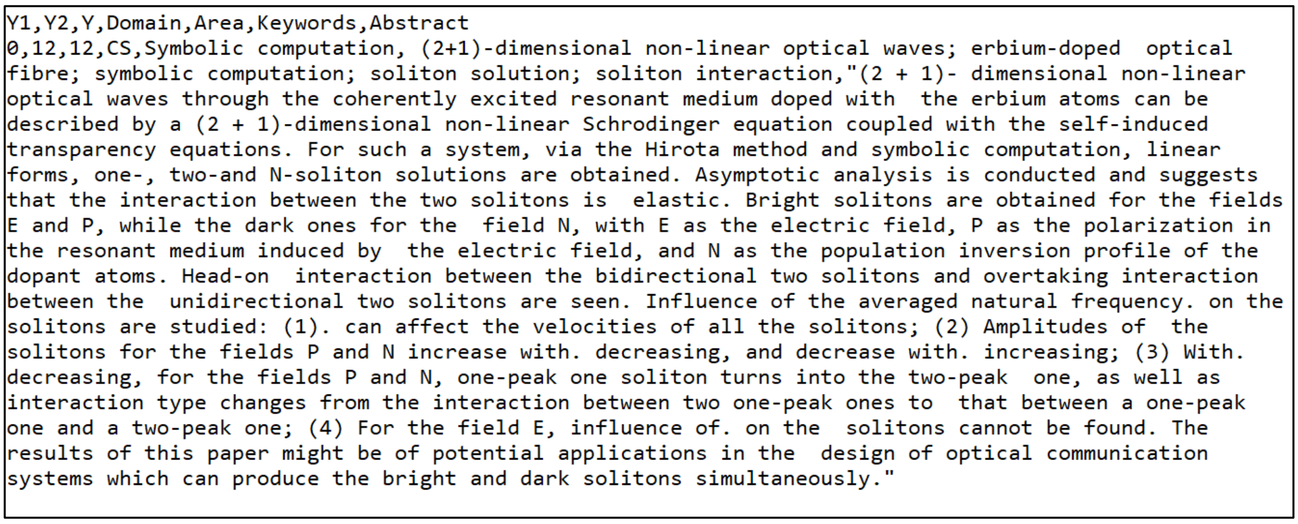
The dataset 1 contains user opinions about the applications in the Google Play Store (see Figure 6). There are two files in this dataset. We only used the text-heavy one. The file has five attributes: App (application name), Translated_Review (user opinion translated from different languages), Sentiment (positive or negative opinion), Sentiment_Polarity, Sentiment_Subjectivity.



```
App,Translated_Review,Sentiment,Sentiment_Polarity,Sentiment_Subjectivity
10 Best Foods for You,nan,nan,nan,nan
10 Best Foods for You,"I like eat delicious food. That's I'm cooking food myself,   case ""10
Best  Foods"" helps lot, also ""Best Before (Shelf   Life)""",Positive,1,0.533333333
```

Figure 6. A screenshot example from dataset 1

The dataset 2 provides information on 46,985 documents with 134 categories, including seven Web of Science parent categories (see Figure 7).

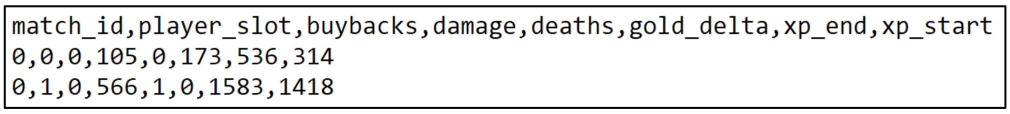


```
Y1,Y2,Y,Domain,Area,Keywords,Abstract
0,12,12,CS,Symbolic computation, (2+1)-dimensional non-linear optical waves; erbium-doped  optical
fibre; symbolic computation; soliton solution; soliton interaction,"(2 + 1)- dimensional non-linear
optical waves through the coherently excited resonant medium doped with  the erbium atoms can be
described by a (2 + 1)-dimensional non-linear Schrodinger equation coupled with the self-induced
transparency equations. For such a system, via the Hirota method and symbolic computation, linear
forms, one-, two-and N-soliton solutions are obtained. Asymptotic analysis is conducted and suggests
that the interaction between the two solitons is  elastic. Bright solitons are obtained for the fields
E and P, while the dark ones for the  field N, with E as the electric field, P as the polarization in
the resonant medium induced by  the electric field, and N as the population inversion profile of the
dopant atoms. Head-on  interaction between the bidirectional two solitons and overtaking interaction
between the  unidirectional two solitons are seen. Influence of the averaged natural frequency. on the
solitons are studied: (1). can affect the velocities of all the solitons; (2) Amplitudes of  the
solitons for the fields P and N increase with. decreasing, and decrease with. increasing; (3) With.
decreasing, for the fields P and N, one-peak one soliton turns into the two-peak  one, as well as
interaction type changes from the interaction between two one-peak ones to  that between a one-peak
one and a two-peak one; (4) For the field E, influence of. on the  solitons cannot be found. The
results of this paper might be of potential applications in the  design of optical communication
systems which can produce the bright and dark solitons simultaneously."
```

Figure 7. A screenshot example from dataset 2

Following are features of the dataset 2 as shown in Figure 7: Y1 (target value), Y2 (target value of level one-parent label), Y (target value of level one-child label), Domain (including seven primary domains: computer science, electrical engineering, psychology, mechanical engineering, civil engineering, medical science, and biochemistry), Area (subdomain), Keywords (papers' keywords), Abstract (include text sequences of published papers).

The dataset 3 includes 50000 ranked ladder matches from the Dota 2 data dump generated by Opendota (see Figure 8). There are 19 files in the dataset, but only teamfights_players.csv is used in this study. The file has eight properties: match_id (individual player ids), player_slot (link to other files in the dataset), buybacks, damage, deaths, gold_delta (status of winning or losing gold), xp_end (experience at the end of the game), xp_start (gain experience early in the game).



```
match_id,player_slot,buybacks,damage,deaths,gold_delta,xp_end,xp_start
0,0,0,105,0,173,536,314
0,1,0,566,1,0,1583,1418
```

Figure 8. A screenshot example from dataset 3

4.3. Queries

In this section, the indexing and search queries that will be utilized in the study are explained. After explaining the file structures employed by Apache Solr and Elasticsearch, the indexing queries are examined. Finally, search queries were explained in detail. Moreover, there are restrictions on the kind of files that may be indexed by Elasticsearch indexing queries. It is possible to index just New Delimited JSON (ND-JSON) files using curl. There is no such restriction with Apache Solr; JSON, CSV, and XML files may also be indexed using a curl query. However, in order to give a more accurate comparison with Elasticsearch, it was determined that Apache Solr would also employ a JSON file. Figure 9 provides a simple two-document JSON file showing the general structure of JSON files to be indexed in Apache Solr.

```
[
    {
        "App": "10 Best Foods for You",
        "Translated_Review": "nan",
        "Sentiment": "nan",
        "Sentiment_Polarity": "nan",
        "Sentiment_Subjectivity": "nan"
    },
    {
        "App": "10 Best Foods for You",
        "Translated_Review": "I like eat delicious food. That's I'm cooking food
myself, case '10 Best Foods' helps lot, also 'Best Before (Shelf Life)'",
        "Sentiment": "Positive",
        "Sentiment_Polarity": "1",
        "Sentiment_Subjectivity": "0.533333333"
    }
]
```

Figure 9. A simple example for JSON

Figure 10 depicts the two-document structure of Elasticsearch-generated ND-JSON files. In contrast to the JSON file, the content does not begin and stop with [ ] brackets. Additionally, lines containing index information ({“index”: {...}}) must be included in each document. ND-JSON is a collection of JSON items that are separated by “\n”. Creating ND-JSON also requires cost and time.

```
{"index": {"_index": "apps_reviews", "_id": "1"}
{"App":"10 Best Foods for You", "Translated_Review":"nan", "Sentiment":"nan",
"Sentiment_Polarity":"nan", "Sentiment_Subjectivity":"nan"}
{"index": {"_index": "apps_reviews", "_id": "2"}
{"App":"10 Best Foods for You", "Translated_Review":"I like eat delicious food. That's I'm
cooking food myself, case '10 Best Foods' helps lot, also 'Best Before (Shelf Life)'",
"Sentiment":"Positive", "Sentiment_Polarity":"1", "Sentiment_Subjectivity":"0.533333333"}
```

Figure 10. A simple example for ND-JSON

Apache Solr runs on port 8983 on the local computer, and Elasticsearch runs on port 9200. As seen in Table 4, the general structures of indexing queries for both technologies are given. In Apache Solr, the core name for the dataset to be processed is written in the **core_name** field, while in Elasticsearch it is written in the **indice_name** field. The core/indice names used in the query structure are apps_reviews for dataset 1, wos_papers for dataset 2, and teamfights_players for dataset 3. Subsequently, for the files to be indexed in both Apache Solr and Elasticsearch, a file named example was created in their respective directories. In the file posted for indexing, the json file for Apache Solr is named with core_name while in Elasticsearch the nd-json file is generated using **nd_** per indice_name. For example, the curl query for dataset 1 will be core/indice name apps_reviews, while the file to be posted to Apache Solr will be apps_reviews.json, and the file to be posted to Elasticsearch will be nd_apps_reviews.json. In addition, the indexing query performs the indexing process by specifying the type of the query as POST and showing the file path.

Table 4. General structures for indexing queries

Technology	**Indexing query**
Apache Solr	curl -H "Content-Type: application/json" -XPOST http://localhost:8983/solr/**core_name**/update -T "C:/solr-9.0.0/example/**core_name**.json"
Elasticsearch	curl -H "Content-Type: application/x-ndjson" -XPOST http://localhost:9200/_bulk --data-binary @C:/elasticsearch-8.4.2/example/nd_**indice_name**.json

In Table 5, a general query format for Apache Solr and Elasticsearch search operations is provided. The query content element is shared by both technologies. Elasticsearch also uses the Kibana Query Language (KQL) format, but the Lucene query language, which is supported by both platforms, is utilized to monitor search times more accurately. In addition, the Elasticsearch query's **track_total_hits** option, which is false by default, was changed to true. Because Apache Solr detects all results during a search, Elasticsearch does not focus on the remaining data once it has found a particular threshold. As with Apache Solr, setting the **track_total_hits** argument is essential for Elasticsearch to locate all hits. The default value of the **track_total_hits** option in Elasticsearch is 10,000.

Table 5. General structures of the searching queries

Technology	Searching query
Apache Solr	http://localhost:8983/solr/**core_name**/select?q=**query content**
Elasticsearch	http://localhost:9200/**indice_name**/_search?q=**query content**&track_total_hits=true

Table 6 demonstrates that a total of 10 queries with varying levels of complexity were executed: Q1-Q4 applied for the dataset 1, Q5-Q7 prepared for the dataset, and Q8-Q10 applied for the dataset 3. Each data set's queries were constructed in a manner that increases query complexity. AND and OR are utilized to query several fields. In the Lucene Query Language, the symbol * indicates that the word might be an expression at the beginning or end, whereas the expression [number1 TO number2] specifies values between the numbers number1 and number2.

Table 6. Searching queries used for the three datasets

Dataset	Query	Purpose	query content
Dataset 1	Q1	List Translated_Review values equal to ‘nan’	Translated_Review:nan
	Q2	List Sentiment_Subjectivity value between 0.1 and 0.746	Sentiment_Subjectivity:[0.1 TO 0.746]
	Q3	List App value equal to ‘Food’ and Sentiment value equal to ‘Positive’ and Translated_Review value equal to ‘Full’ or ‘great’ or ‘good’ or ‘enjoy’	App:*Food* AND Sentiment:Positive AND Translated_Review:*Full* OR Translated_Review:*great* OR Translated_Review:*good* OR Translated_Review:*enjoy*
	Q4	List Sentiment_Subjectivity value between 0.79 and 0.82 and Sentiment_Polarity value equal to 0.716666667	Sentiment_Subjectivity:[0.79 TO 0.82] AND Sentiment_Polarity:0.716666667
Dataset 2	Q5	List Keywords value equal to ‘Parkinson’	Keywords:*Parkinson*
	Q6	List Keywords value equal to ‘algorithm’ or Abstract value equal to ‘algorithm’	Keywords:*algorithm* OR Abstract:*algorithm*
	Q7	List Keywords value equal to ‘analysis’ or Domain value equal to ‘CS’ or Abstract value equal to ‘system’	Keywords:analysis OR Domain:*CS* OR Abstract:system
Dataset 3	Q8	List xp_end value equal to 32417	xp_end:32417
	Q9	List buybacks value equal to 1 and deaths value equal to 1	buybacks:1 AND deaths:1
	Q10	List buybacks value equal to 0 and deaths value equal to 1 and damage value equal to 0 or gold_delta value 0	buybacks:0 AND deaths:1 AND damage:0 OR gold_delta:0

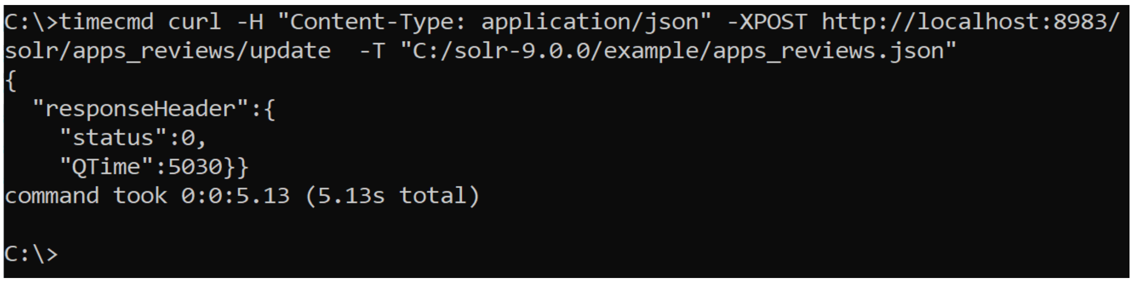
5. Results and evaluation

This section compares and contrasts Apache Solr with Elasticsearch's indexing and search capabilities. Then, every aspect of the comparison is described in depth.

5.1. Indexing

In this section, a comparison of Apache Solr and Elasticsearch regarding indexing is performed on three datasets utilizing three different hardware configurations (see Table 2). First, the difference in indexing times for the default heap size is measured. Then, the indexing times for different file sizes on the most efficient machine (Machine 1) are compared based on and different heap sizes in GB (6, 8, 12, 16, 20, and 24) as shown in Table 8 to learn more about how heap size affects indexing.

As shown in Table 4, indexing performance is assessed using curl requests on each system via the Windows command line. In Figure 11, the output of a query run on command prompt is shown. To calculate the indexing time, the timecmd command is added at the beginning of the Apache Solr and Elasticsearch curl queries given in Table 4. This command refers to a batch file that calculates the runtime of curl queries. Command took is the time calculated by the timecmd batch script. In this study, command took times are considered.

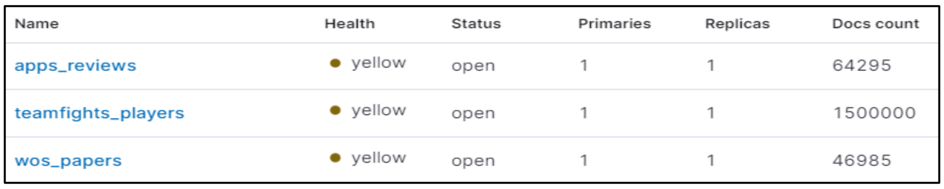


```
C:\>timecmd curl -H "Content-Type: application/json" -XPOST http://localhost:8983/
solr/apps_reviews/update  -T "C:/solr-9.0.0/example/apps_reviews.json"
{
  "responseHeader":{
    "status":0,
    "QTime":5030}}
command took 0:0:5.13 (5.13s total)

C:\>
```

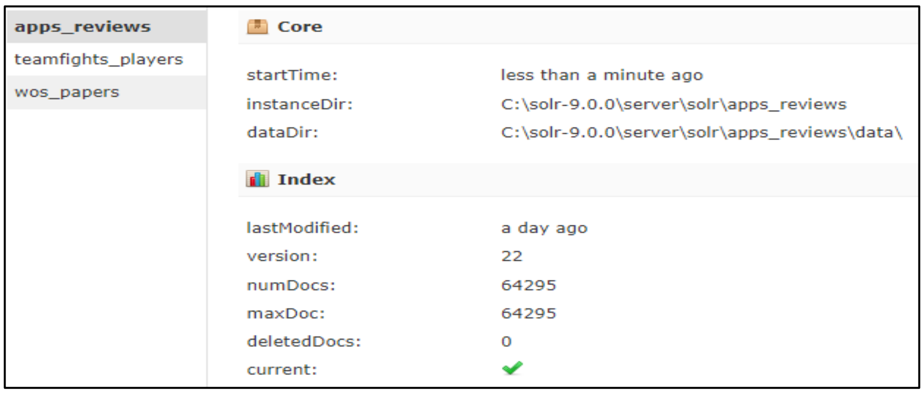
Figure 11. A curl query submitted in the command prompt

After the indexing queries given in Table 4 that have been run for the three datasets, Figure 12 presents information (health status, number of replicas, count of documents) about the three indices and indices created in Elasticsearch. This is a screenshot taken using Kibana. Likewise, Figure 13 shows the cores created in Apache Solr and some details (the paths where the core is created and the data is found, the number of documents, the core's active status, log dates).



Name	Health	Status	Primaries	Replicas	Docs count
apps_reviews	● yellow	open	1	1	64295
teamfights_players	● yellow	open	1	1	1500000
wos_papers	● yellow	open	1	1	46985

Figure 12. Indices created in Elasticsearch



apps_reviews
teamfights_players
wos_papers

Core

startTime:	less than a minute ago
instanceDir:	C:\solr-9.0.0\server\solr\apps_reviews
dataDir:	C:\solr-9.0.0\server\solr\apps_reviews\data\

Index

lastModified:	a day ago
version:	22
numDocs:	64295
maxDoc:	64295
deletedDocs:	0
current:	✓

Figure 13. Cores created in Apache Solr

Table 7, on the default heap size, displays the indexing times for Apache Solr and Elasticsearch based on the size of the datasets. Apache Solr's heap size is 512 MB by default, but Elasticsearch's heap size is half of its RAM capacity. In addition, for each curl query, each indexing request was executed five times, and the average values were computed and averaged. Before each indexation attempt, Apache Solr and Elasticsearch Tools indexes and data are removed and parameters are reset to default.

Table 7. Indexing times (sec) for default heap size

	Machine 1		**Machine 2**		**Machine 3**	
Dataset size	**Apache Solr**	**Elasticsearch**	**Apache Solr**	**Elasticsearch**	**Apache Solr**	**Elasticsearch**
~17 MB	2.004	3.445	2.46	4.386	5.968	11.852
~75 MB	4.36	6	5.818	6.382	11.758	20.922
~300 MB	33.012	72.752	44.426	81.37	89.028	140.788

Figure 14 illustrates a graph of the average indexing times listed in Table 7. Apache Solr outperforms Elasticsearch on used computers in terms of average indexing speeds across three distinct datasets. As seen in Figure 14, indexing time increases as dataset size grows.

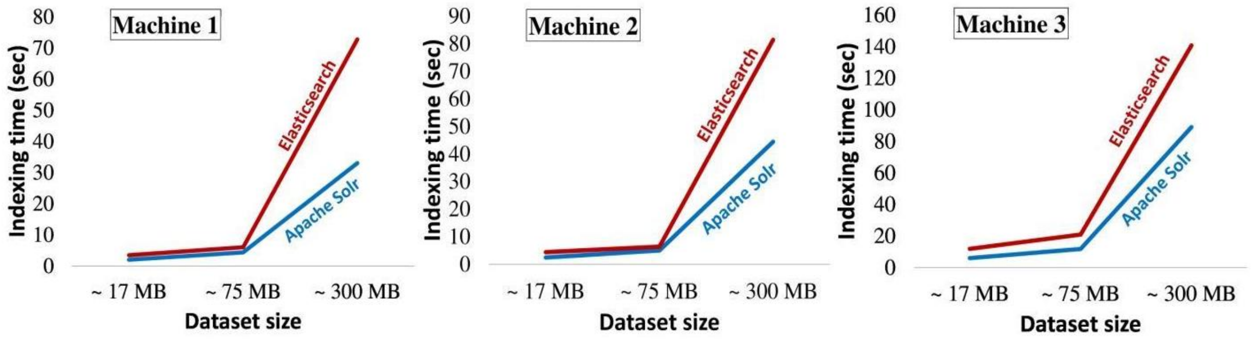


Figure 14. Comparison of indexing times (sec) regarding different machines

Figure 15 illustrates the relationship between the indexing times on various computers and the heap size for Apache Solr and Elasticsearch technologies. Machine 1 is faster at indexing than the other two computers since it has more accessible system resources (see Table 2). Therefore, powerful computers' capabilities are advantageous for indexing. However, it would be misleading to assert that the most efficient machine is necessarily the best. By adjusting the heap size, the indexing performance of the two tools is now similar in the continuing of this study.

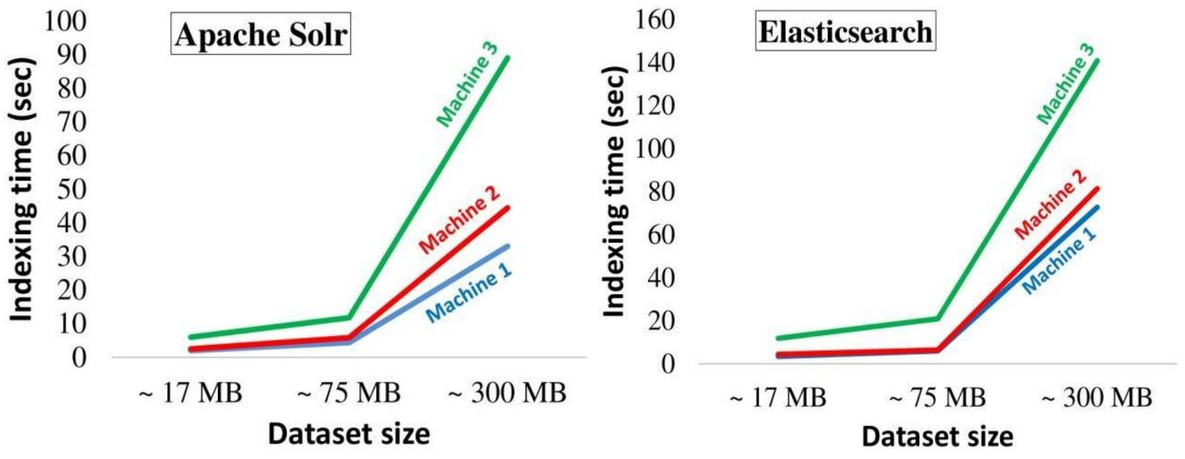


Figure 15. Indexing performance of both technologies on different machines

Table 8 illustrates the indexing times for six different heap sizes on Machine 1, which has the best computer resources. These are increment values that were picked at random. The objective is to compare the performance of both tools based on the size of the heap. In this context, the reindexing times for the three datasets for the two technologies are measured.

Table 8. A comparison of indexing times (sec) based on heap size

	Apache Solr			Elasticsearch		
Heap size	**Dataset 1**	**Dataset 2**	**Dataset 3**	**Dataset 1**	**Dataset 2**	**Dataset 3**
6 GB	1.976	4.574	31.788	2.768	5.398	70.106
8 GB	1.824	4.486	32.32	2.4	4.41	47.148
12 GB	1.788	4.51	32.072	3.478	6.266	70.908
16 GB	1.808	4.698	32.104	2.648	5.208	62.638
20 GB	1.868	4.772	32.528	2.396	4.378	48.038
24 GB	2.198	4.942	32.176	2.756	5.494	65.986

In Figure 16, the two technologies' indexing performances are compared regarding the size of the heap. Considering the performance of Apache Solr, it is seen that it is not as dependent on machine features as Elasticsearch. Increasing the heap size does not seem to noticeably affect the indexing performance of Apache Solr. In contrast, the indexing performance has changed for Elasticsearch due to an increase in heap size. Moreover, the performance of Elasticsearch is not linear in terms of the linearly raised heap size. It shows better results at 8 GB and 20 GB. In this situation, too large a heap size does not always give the best performance. In this case, there is no ideal or fixed size for manually setting the heap size. Developers need to experiment and find a way that fits their work environment and the purpose of the project. Moreover, according to experts, heap size should not exceed 50% of total memory. They also recommend keeping the heap size at its default for Elasticsearch. Furthermore, while indexing, Elasticsearch prints the data to the command screen; this situation has a slight negative impact on search time.

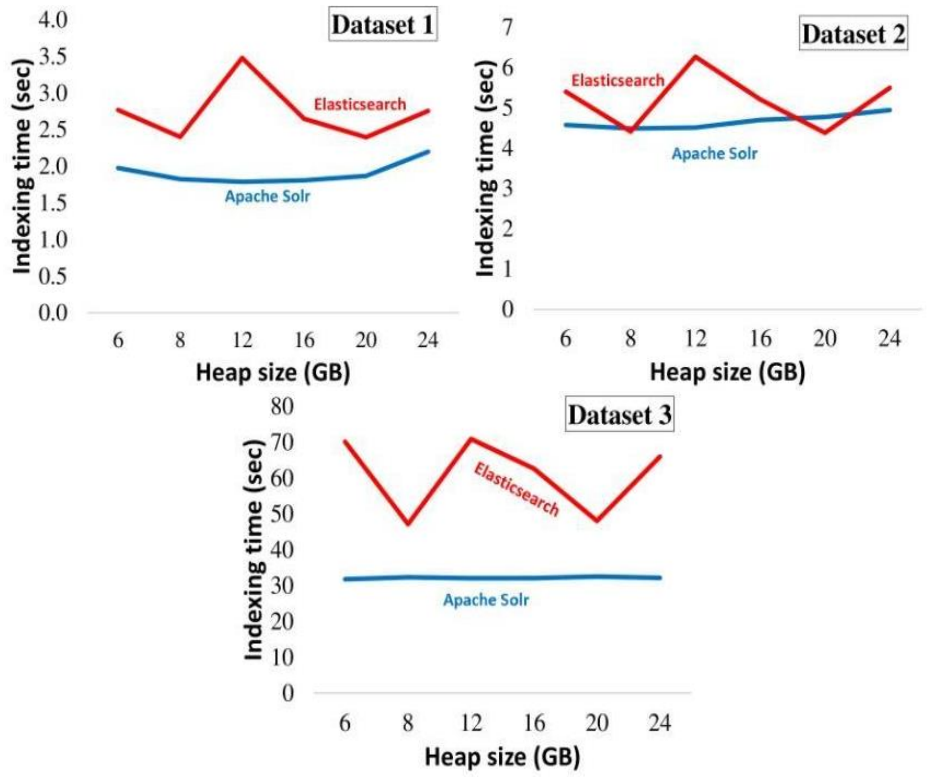
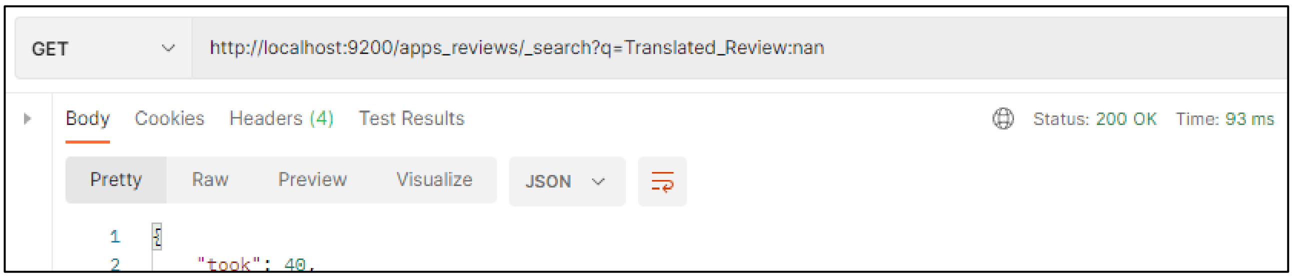


Figure 16. Change of indexing time according to machine heap size

As a result, according to our results, Apache Solr performs better for indexing than Elasticsearch. We can also say that, when it comes to indexing, hardware resources are more important for Elasticsearch than for Apache Solr.

5.2. Searching

In this study, Postman, an API platform for creating, constructing, and testing APIs, is utilized to monitor search times. The queries (see Table 6) are provided as a request, and responses are retrieved from Postman. It took ten attempts for each query to be answered, and the mean values of all were added to the tables. In addition, Figure 17 shows the Postman interface and a submitted search request. URL requests were performed using the GET method. Time found on the right side of the screenshot shows the search times used in this study.


GET http://localhost:9200/apps_reviews/_search?q=Translated_Review:nan

Status: 200 OK Time: 93 ms

```
{
    "took": 40,
```

Figure 17. A screenshot of the Postman interface

Table 9 displays the average search time for each query and the number of records obtained for the Apache Solr and Elasticsearch technologies when each query is submitted ten times. Pseudocode for a query is as follows:

```
1. Create query string (based on user requests)
2. Submit query request via Postman
3. Get searching time
4. Receive results
```

Changing the default value of the track_total_hits parameter in Elasticsearch reveals differences in search time. The field type specifies the data type of each query field. During the search, the highest number of received data points was 55263 in Q10. The track total hits level is therefore set to 60000, which accounts for the quantity of data points returned by all queries. This threshold was set to 60000 in our example for testing purposes; however, this quantity may be altered based on the datasets employed and expected outcomes.

Table 9. Comparison of searching times (sec)

Query	**Apache Solr**	**Elasticsearch** / track_total_hits			
		= True	**= False**	**Field types**	**Result (count)**
Q1	0.076	0.097	0.107	Text	26863
Q2	0.058	0.117	0.09	Numeric	26866
Q3	0.205	0.21	0.199	Text	6
Q4	0.064	0.081	0.089	Numeric	6
Q5	0.114	0.147	0.139	Text	294
Q6	0.141	0.31	0.298	Text	3955
Q7	0.089	0.09	0.117	Text	14926
Q8	0.048	0.073	0.069	Numeric	237
Q9	0.051	0.107	0.099	Numeric	26089
Q10	0.06	0.209	0.136	Numeric	55263

The searching times shown in Table 9 indicate that total hits were determined for both technologies, but only 10 values were printed. Thus, a fair comparison is achieved. While Q3 and Q7 queries have approximately the same search time for both technologies, Q6 and Q10 queries have a significant difference. As a result, based on our datasets and the queries, overall, Apache Solr performs better than Elasticsearch in demanding searching jobs. Moreover, as shown in Table 9, despite the fact that the returned results for Q1 and Q2 are many times greater than the query Q3, the retrieval times for Q1 and Q2 are faster than Q3. The fact that Q3 is a more complex query (see Table 6) causes the search time to increase, while it is seen that the number of results and search times are not directly proportional. In another case, Q3 and Q4 return 6 results, but for both Apache Solr and Elasticsearch, Q4 has a better search time than Q3. When both queries given in Table 6 are examined in the same dataset, the fields in Q3 have text content, while the fields in Q4 have numerical data. Thus, in this case, the search time has also been significantly impacted by the data and field types. Figure 18 provides the graphical representation of the query results presented in Table 9.

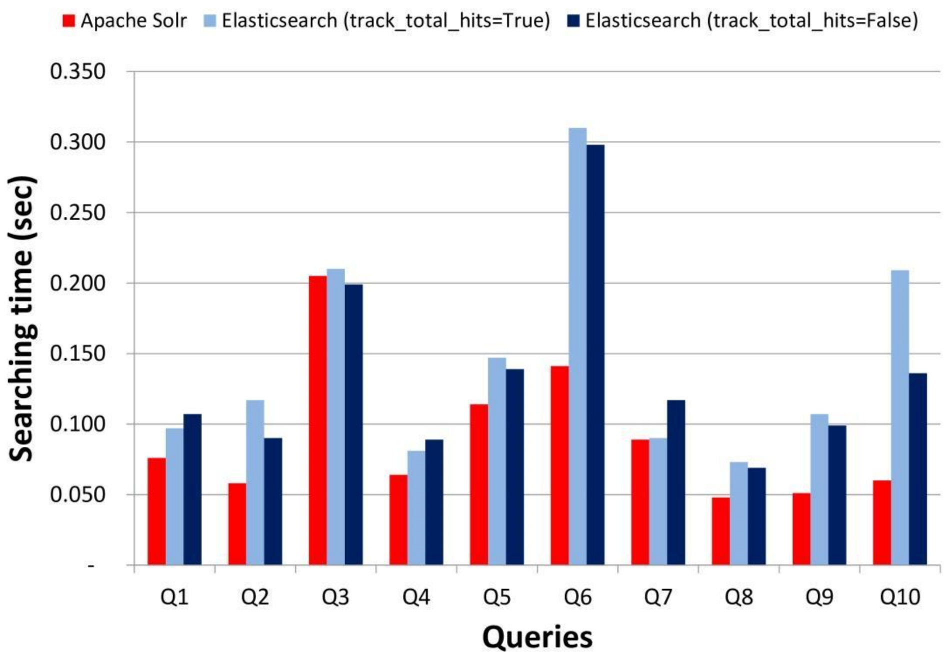


Figure 18. Results of searching times regarding to prepared queries

In Figure 18, if the track_total_hits parameter is false in Elasticsearch, it causes a decrease in the search time, except for the queries Q1, Q4, and Q7. A more detailed search for queries Q1, Q4, and Q7 may allow for better results, or increasing the number of trials for each query in this study may improve the search time for these three queries. As a result, setting Elasticsearch's track_total_hits parameter to true or false also significantly impacts the search speed. To sum up, despite these two situations, Apache Solr overall performs better than Elasticsearch in terms of searching times.

6. Discussion and future work

In our investigation, we performed several comparisons between Apache Solr and Elasticsearch. In terms of technical features, both technologies are significant and lead in terms of full-text search. Then, we compared the indexing and search speeds of Apache Solr and Elasticsearch based on the queries and indexing times described in Section 5. In terms of indexing speeds, according to our findings, Solr is superior to Elasticsearch. In addition, we observed that improving the hardware capabilities of the system utilized for Elasticsearch is more critical. Based on a comparison of search times, Apache Solr performs better than Elasticsearch. Moreover, the fact that both solutions are built on Apache Lucene demonstrates that they share several robust characteristics.

Apache Solr is a project that has been established by a strong community of Apache Software Foundation. This brought features such as faceting search, feature filtering, real-time analytics to Apache Solr in this development process. Apache Solr is also more advantageous than Elasticsearch, which was released four years after Apache Solr, when it comes to project development based on user feedback. Because it provided an opportunity to take the project to a more mature stage. Nowadays, both technologies have reached a certain level. This study demonstrates that Apache Solr is more pro-user in terms of usage and search functionality. Because it has developed into a simpler and clearer structure. Moreover, Apache Solr seems to be good at both indexing and searching. While hardware specifications are important for Elasticsearch, Apache Solr performs better with the same hardware specifications. This indicates that Apache Solr is more stable.

Furthermore, we present insight comparisons between our work and previous studies on the performance comparison of Apache Solr and Elasticsearch. Table 10 contains comparison parameters and insight for experiences similar to this study. While studies Yurtsever et al. (2022) and D.S. (2016) have the same opinion as our study for indexing performance, studies Elasticsearch vs. Solr Performance: Round 2 (2015), Hansen et al. (2018), and Gonçalves and Sunye (2020) indicate that Elasticsearch has a better indexing time in terms of indexing performance. Considering the searching times, Apache Solr is better in terms of queries per second in Elasticsearch vs. Solr Performance: Round 2 (2015). On the other hand, according to Hansen et al. (2018), and D.S. (2016), Elasticsearch is better at searching. At the same time, we also observed similar results as in Hansen et al. (2018) about Elasticsearch's use of virtual memory. In addition, some of the datasets in the related literature were not accessible, and some were not appropriate for this study. Therefore, the numerical values could not be compared. However, future studies may use the datasets we used in this study.

Table 10. Comparison of regarding related works

Paper	Comparison criteria	Compared technologies	Insight
Elasticsearch vs. Solr Performance: Round 2 (2015)	Search test with indexing, search test with indexing load, queries per second (OPS) test	Apache Solr vs. Elasticsearch	Regarding indexing time, Elasticsearch is good on small data whereas Solr is better on large data. Elasticsearch is good to test with indexing load. Solr is good on the QPS test.
Hansen et al. (2018)	Indexer and searcher performance, use virtual machine	Apache Solr vs. Elasticsearch	Elasticsearch is better for index size and indexing time. In search, first run Elasticsearch is good, second run Solr is better. Elasticsearch uses more virtual memory.
Yurtsever et al. (2022)	Insert times	Apache Solr vs. Elasticsearch	Apache Solr is faster and better.
D.S. (2016)	Indexing speed, search speed	Apache Solr, Elasticsearch, Sphinx, Xapian	Apache Solr is good for indexing, while Elasticsearch is good for searching.
Gonçalves and Sunye (2020)	Indexing time, RAM usage, index disk space	Apache Solr vs. Elasticsearch	Generally, Elasticsearch performs better than Apache Solr.

In this study, we evaluated the performance of Apache Solr and Elasticsearch on individual machines. In the future, we want to do this work in a distributed environment to develop multi-node Apache Solr and Elasticsearch in order to evaluate the indexing and searching performance of both technologies in a distributed environment in order to determine the various aspects of this study. In this way, we will clearly see how the distributed indexing performance and the search process on these distributed data compare to the performance in a single node.

7. Conclusion

In conclusion, we provide a comprehensive report on full-text search methods used for processing and analyzing large amounts of data. Initially, the features and technical comparison of Apache Solr and Elasticsearch technologies are provided in depth. Second, a comprehensive and fair comparison of both systems is conducted based on indexing times and carefully designed queries on three separate textual and numeric datasets. Furthermore, a detailed related work is provided to present similarities, differences, and our insights on how to use Apache Solr and Elasticsearch technologies in terms of underlying hardware. According to our results, in general Apache Solr performs better than Elasticsearch. We suggest using Apache Solr with computers that have low hardware resources. Our work offers researchers with background information on full-text search methods, and our findings shed light on selecting the optimal alternative for full-text searching activities in terms of accessible hardware sources, data type, and data size.

Author contribution

Ayşenur Deniz has worked on all stages of writing, coding, making figures, studying related works, collecting querying results, revising, and comparing technologies on different machines. Muhammed Mehdi Elömer has worked at writing, studying related works, and collecting querying results. Ahmet Arif Aydin has worked on entire stages of writing, editing, revising, and consulting with co-authors.

Declaration of ethical code

The authors of this article declare that the materials and methods used in this study do not require ethical committee approval or special legal permission.

Conflicts of interest

The authors declare that they do not have any conflict of interest.

References

Anderson, K. M., Aydin, A. A., Barrenechea, M., Cardenas, A., Hakeem, M., & Jambi, S. (2015). Design challenges/solutions for environments supporting the analysis of social media data in crisis informatics research. *2015 48th Hawaii International Conference on System Sciences, 2015-March,* 163–172. https://doi.org/10.1109/HICSS.2015.29

Apache Lucene. (2022). https://lucene.apache.org/

Barrenechea, M., Jambi, S., Aydin, A. A., Hakeem, M., & Anderson, K. M. (2017). Getting the query right for crisis informatics design issues for web-based analysis environments. *Journal of Web Engineering*, *16*(5), 399–432. https://journals.riverpublishers.com/index.php/JWE/article/view/3269/2153

Bellini, P., Bugli, F., Nesi, P., Pantaleo, G., Paolucci, M., & Zaza, I. (2019). Data flow management and visual analytic for big data smart city/IOT. *Proceedings - 2019 IEEE SmartWorld, Ubiquitous Intelligence and Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Internet of People and Smart City Innovation, SmartWorld/UIC/ATC/SCALCOM/IOP/SCI 2019*, 1529–1536. https://doi.org/10.1109/SmartWorld-UIC-ATC-SCALCOM-IOP-SCI.2019.00276

DB-Engines. (2022). https://db-engines.com/en/

Domo Company. (2022). *Data Never Sleeps 9.0*. https://www.domo.com/learn/infographic/data-never-sleeps-9

Dota 2 Matches. (2022). https://www.kaggle.com/datasets/devinanzelmo/dota-2-matches

D.S., S. (2016). A quick search on the projects with a high loads and a large amount of data. *Modern Technologies: Current Issues, Achievements and Innovations — Collection of Articles III International Scientific Conference / under the General Editorship of G. Yu Gulyaev — Penza MCNS « Science and Education »*, 23–32.

Elastic Installation and Upgrade Guide [8.4]. (2022). https://www.elastic.co/guide/en/elastic-stack/8.4/index.html

Elasticsearch vs. Solr Performance: Round 2. (2015). https://www.flax.co.uk/blog/2015/12/02/elasticsearch-vs-solr-performance-round-2/

Gonçalves, A. A. S., & Sunye, M. S. (2020). Comparison of search servers for use with digital repositories. *ICEIS 2020 - Proceedings of the 22nd International Conference on Enterprise Information Systems, 1*, 256–260. https://doi.org/10.5220/0009577102560260

Google Play Store Apps. (2022). https://www.kaggle.com/datasets/lava18/google-play-store-apps

Google Trends. (2022). https://trends.google.com/trends/

Halevi, G., & Moed, H. (2012). The evolution of big data as a research and scientific topic: overview of the literature. *Research Trends*, *30*(36), 3–6.

Hansen, J., Porter, K., Shalaginov, A., & Franke, K. (2018). Comparing open source search engine functionality, efficiency and effectiveness with respect to digital forensic search. *NISK 2018 - 11th Norwegian Information Security Conference,* 108-121.

Kılıç, U., & Karabey, I. (2016). Comparison of solr and elasticsearch among popular full-text search engines and their security analysis. *UBMK'16 - International Conference on Computer Science and Engineering, 2016 October*. https://doi.org/10.13140/RG.2.2.24563.32803

Kowsari, K., Brown, D., Heidarysafa, M., Meimandi, K. J., Gerber, M., & Barnes, L. (2018). Web of science dataset. 6. *Mendeley Data.* https://doi.org/10.17632/9RW3VKCFY4.6

Lakhara, S., & Mishra, N. (2017). Desktop full-text searching based on Lucene: a review. *2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)*, 2434–2438. https://doi.org/10.1109/ICPCSI.2017.8392154

Lashkaripour, Z. (2020). The era of big data: a thorough inspection in the building blocks of future generation data management. *International Journal of Scientific and Technology Research*, *9*, 321–330.

Lokoč, J., Veselý, P., Mejzlík, F., Kovalčík, G., Souček, T., Rossetto, L., Schoeffmann, K., Bailer, W., Gurrin, C., Sauter, L., Song, J., Vrochidis, S., Wu, J., & Jónsson, B. þóR. (2021). Is the reign of interactive search eternal? findings from the video browser showdown 2020. *ACM Transactions on Multimedia Computing, Communications, and Applications*, *17*(3), 1–26. https://doi.org/10.1145/3445031

Luburić, N., & Ivanovic, D. (2016). Comparing Apache Solr and Elasticsearch search servers. *6th International Conference on Information Society and Technology - ICIST 2016.* http://www.eventiotic.com/eventiotic/files/Papers/URL/icist2016_54.pdf

Oussous, A., & Benjelloun, F. (2022). A comparative study of different search and indexing tools for big data. *Jordanian Journal of Computers and Information Technology*, *8*(1), 1. https://doi.org/10.5455/jjcit.71-1637097759

Rao, T. R., Mitra, P., Bhatt, R., & Goswami, A. (2018). The big data system, components, tools, and technologies: a survey. *Knowledge and Information Systems*, *60*(3), 1165. https://doi.org/10.1007/s10115-018-1248-0

Resources Apache Solr. (2022). https://solr.apache.org/resources.html

Voit, A., Stankus, A., Magomedov, S., & Ivanova, I. (2017). Big data processing for full-text search and visualization with elasticsearch. *IJACSA - International Journal of Advanced Computer Science and Applications*, *8*(12). www.ijacsa.thesai.org

Wang, J.-F., Wang, X.-F., & Li, H. (2022). Design of multimedia distance teaching auxiliary system based on MOOC platform. *ICMTMA 2022 - 14th International Conference on Measuring Technology and Mechatronics Automation,* 1179–1186. https://doi.org/10.1109/ICMTMA54903.2022.00237

Y. Aldailamy, A., Abdul Hamid, N. A. W., & Abdulkarem, M. (2018). Distributed indexing: performance analysis of Solr, Terrier and Katta information retrievals. *Malaysian Journal of Computer Science*, 87–104. https://doi.org/10.22452/mjcs.sp2018no1.7

Yurtsever, M. M. E., Özcan, M., Taruz, Z., Eken, S., & Sayar, A. (2022). Figure search by text in large scale digital document collections. *Concurrency and Computation: Practice and Experience*, *34*(1). https://doi.org/10.1002/CPE.6529