

The Analysis of Open Source Search Engines

Xinyuan Chang^a

University College, University of Toronto, Toronto, M5S 2E8, Canada.

^aceliachang0519@gmail.com

Abstract. In this information era of high-speed Internet development, the volume of information has grown exponentially along with the expansion in Internet usage by the general people and the steady strengthening of business and institutional information technology. How to find the needed information quickly and accurately from the vast amount of digital information has become an urgent demand of the people of the whole country. The term "open source search engine" simply denotes that anyone may use and alter the source code as they see fit. Open source search engines provide an excellent way and material for people to learn, study and master search technology. This study aims to compare different open source search engines. In this paper compares MeiliSearch and Apache Solr; ElasticSearch and Typesense. Introduce how these engines work and the advantage and disadvantage of these open source search engines. The section four gives a case study of Apache Lucene.

Keywords: Open source search engine, MeiliSearch, ElasticSearch, Apeche Lucene.

1. Introduction

The common search engines could be divided into three broad categories, which are searching for websites including Google, Yandex, Yahoo and Baidu; social networking such as WeChat, Facebook and Instagram, and Social media, for example, Netflix, Youtube and Tiktok. However, there are some limitations that commercial search engines would have. They are close sourced, hard to Customize Searching features and have fewer functions for developers.

1.1. Introduction of open source software

The majority of users of open source software (word processors, spreadsheets, graphics editors, audio editors, etc.) have never made any changes to the source code. Instead, users merely run the provided form on their computers. Additionally, it is faster, more user-friendly for developers, and crucial for businesses in running day-to-day operations and creating analysis projects. Open source search engines promote the popularity and development of search technology while making more and more people begin to understand and promote the use of search technology. The use of open source search engines can significantly reduce the cycle of building search applications and create personalized search applications according to application requirements or even build search engine systems that meet specific needs [1].

This study aims to compare different open source search engines. In this paper, I will compare MeiliSearch and Apache Solr; ElasticSearch and Typesense. A case study of Apache Lucene. The remaining paper is structured as follows: Section 2 introduces how these open source search engines work; section 3 presents the advantages and disadvantages of open source search engines; and section 4 gives a case study.

2. How Open Source Search Engines Work And Feature

2.1. MeiliSearch

MeiliSearch is a RESTful (Representational state transfer) search API that offers a thoroughly prepared solution for anyone looking to give their end consumers a quick and relevant search experience. Additionally, MeiliSearch can support a variety of languages, including English, Japanese,

and Roman. However, it requires a significant time and financial investment, and some small firms employ subpar search engines, which impacts user experience and retention costs.

2.1.1. MeiliSearch's attributes

MeiliSearch can satisfy the necessity of the majority. It provides quick search functions such as typo correction, filters, custom ranks, and more. Following are some of MeiliSearch's main attributes. First, it performs a search while users type, commonly referred to as a "quick search." Researchers can obtain the results by entering the content they wish to query. Additionally, the results displayed will vary as input text content increases. The second feature is that it has filters. On Meilisearch, users can set filters to restrict further and improve the scope of their search results. Then, Meilisearch gives customers the option to sort the query results so they may choose which results to display first. Furthermore, MeiliSearch's association rules offer a straightforward search experience with few options. Even if people enter the wrong word in a search, MeiliSearch can still locate the desired outcomes. Finally, using synonyms helps users simplify and modify their search process. The companies choose to use MeiliSearch are in figure 1 [2].

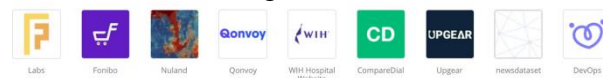


Figure 1. Using MeiliSearch's companies

2.2. Apache Solr

Apache Solr is an open source enterprise search server based on Java, which is easy to expand and modify. It uses the Lucene Java search library for full-text indexing and search. Standard HTTP (Hypertext Transfer Protocol) and XML (Extensible Markup Language) is used for its server connections, but the rest (representational state transfer). JSON (JavaScript object representation) and API (Application Programming Interface) are also supported.

The three steps of Apache Solr's operation are indexing, querying, and ranking the results. Therefore, there are several ways to index documents as a first step. Through index processing, users can directly upload files in JSON, XML/XSLT, or CSV formats to Solr. The Extracting Request Handle, which is the Solr Cell, can be used if we want the file to be in the format of a PDF or an Office document [3]. Tika is used in this request to parse the incoming file and extract the fields that must be indexed to finish the program's processing. Databases, emails, RSS feeds, XML data, plain text files, and other sources can also be used to import data. By using column names as document field names, the DataImport Handler plug-in by Solr may extract and index data from a database. The following step is querying. For instance, people can look for terms like keywords, images, or location information. When they send a query to Solr, Solr can handle and process the request. The handler performs similar tasks to the index handler, but instead of uploading documents, it returns them from the Solr index. The outcomes are ranked as the last step. Solr will arrange the results based on relevance and display the most relevant results at the top for users to view when the search content we provide matches the index document [4].

2.2.1. Apache Solr's features

Similar to other open source search engines, Apache Solr provides a number of features, including full-text searching and support for XML, JSON, and conventional HTTP communication. Additionally, it is extremely fault resistant and scalable. It supports English, German, Chinese, and Japanese, unlike MeiliSearch.

2.3. ElasticSearch

ElasticSearch is a Lucene library and full-text search engine like other open search engines. A distributed search and analysis engine is at the centre of the elastic stack [5]. Managing and retrieving document-oriented semi-structured data is how ElasticSearch operates. Inverted indexing is a technique that quickly locates documents that contain a specific keyword by mapping each distinct

"word" (tag) to a list of documents (locations) that contain it. One or more partitions contain index information (also known as fragments). Elastic search has the ability to replicate and dynamically allocate fragments to cluster nodes [6]. ElasticSearch has numerous investors. ElasticSearch received 10 million from Benchmark, DCVC, and Rod Johnson in November 2012. According to the statistical data, there are three thousand six hundred twenty-two companies use ElasticSearch, including Uber, Shopify, and Robinhood [7].

2.4. Typesense

Typesense is the last open source search engine. With cutting-edge search algorithms that use the most recent advancements in hardware capabilities, Typesense is a modern, open source search engine built from the bottom up. Additionally, it is a more straightforward option than ElasticSearch [8].

3. Advantages And Disadvantages

In order to analyze these open source search engines, people need to understand the advantages and disadvantages of each search engine.

3.1. MeiliSearch benefits and limitations

MeiliSearch's benefits for developers, include its scalability, maintainability, and adaptability. Meilisearch requires barely a few adjustments to launch and operate. For the majority of projects, Meilisearch's default options are sufficient. On the other hand, search continues to be flexible. The restful (representational state transfer) API is used for communication because the majority of developers are already familiar with its definition (Application Programming Interface). Additionally, the search process should be simple for users to concentrate on the search results. Meilisearch aims to offer a straightforward search process with a response time of fewer than fifty milliseconds. Meilisearch additionally expedites and streamlines the search process by letting consumers concentrate on the search results [9].

Despite its many benefits, its limitation should not be overlooked. Each search query examines a maximum of ten terms. Only ten words can be entered in a MeiliSearch query. Even if there are more than ten, they would not be looked up because it could take longer to answer queries with many search terms. Additionally, the default database can only be up to 100 GiB in size. This size can be altered using the configuration reference's options (`—max-index-size` & `—max-task-DB-size`). Meilisearch uses two databases: one for jobs and one for storing data. The amount of space that must be set aside on the disc for both at launch is what LMDB needs to know. Each characteristic of Meilisearch can index up to 65535 places, which is similar to the first restriction. Due to performance and storage difficulties, if it exceeds, it will be disregarded and generate errors. Documents with too many fields generate an internal data structure that is too large, resulting in a large database on disc and slow search performance. In the index, document storage is also constrained. It can hold 4294967296 documents in total. This is the maximum number of documents saved in the index because the internal document identification in Meilisearch's engine uses unsigned numbers. Its payload size is also restricted to 100MB. Most people may benefit from a quick and convenient search experience thanks to the small payload size [10].

3.2. Apache Solr advantage and shortage

Apache Solr offers the benefit of having a rich document parsing system if we design a variety of complex searches. It features speedy reaction times and rich document parsing. The query language's adaptability and power allow it to transform complex queries as well. Additionally, since the cluster mode has a distinct master and slave, people can scale each kind according to the amount of data they need to enter or the response time required [11].

However, it has some limitations. Apache Solr can only be used in private networks since it lacks authentication and authorization. Second, zookeepers are required while using Solr cloud. Most significantly, Solr's master node needs to be reconfigured if it fails. Furthermore, 23 organisations, including Walmart, doubleSlash, and AlphaSense, use Apache Solr as part of their tech stacks.

3.3. Compare MeiliSearch and Apache Solr

MeiliSearch and Apache Solr vary in that many of the most popular websites in the world use Solr for search and navigation. On the other hand, MeiliSearch is specifically described as an "Ultra relevant, instant and typo-tolerant full-text search API." It is an open source search engine that is reliable, speedy, and easy to deploy. Additionally, MeiliSearch is rated far worse than Apache Solr in terms of databases, search APIs, custom search engines, and custom searches [12].

3.4. ElasticSearch advantages

ElasticSearch offers two advantages: it is scalable and resilient. ElasticSearch was created from the ground up to offer constant peace of mind and runs in a distributed setting. Add another node, and our cluster will grow to accommodate your needs. High availability and clustering are further features. A cluster is made up of one or more nodes. When a node fails, ElasticSearch clusters use primary and replica shards to provide failover. The duplicate replaces a destroyed primary shard. The term "node" refers to a single instance of the ElasticSearch process. It is a server for data storage that performs the clustered index and search. Nodes in the cluster can locate one another, thanks to the common cluster name. Horizontal scalability, automatic node recovery, automatic data rebalancing, cross-cluster replication, and cross-datacenter replication are additional features of ElasticSearch [13].

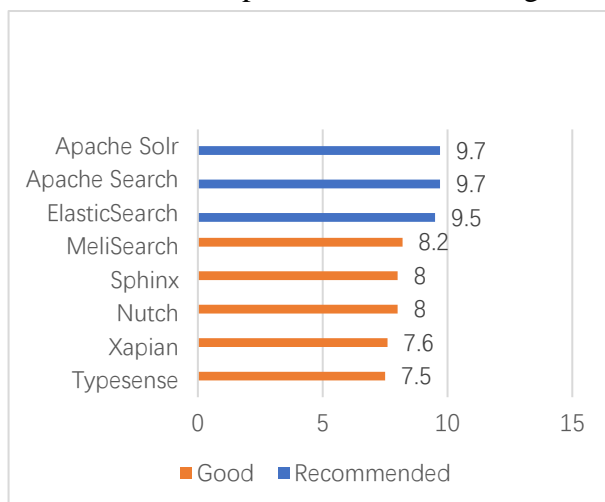
3.5. ElasticSearch and Typesense

This study compares ElasticSearch with Typesense in the content that follows. ElasticSearch is the only source available, but Typesense offers a fully open source. Typesense had its initial commit in 2015, while ElasticSearch made its debut in 2021. Additionally, Typesense uses C++, whereas ElasticSearch uses Java [14]. Moreover, Typesense is the fundamental search algorithm created from scratch. The final one is regarding typo tolerance; ElasticSearch has a poor typo tolerance compared to Typesense [15].

Furthermore, because there are so many configurations to manage, building and maintaining reliable ElasticSearch clusters can be difficult for individuals. However, large teams with sufficient bandwidth benefit more from using this technology. However, Typesense use less set-up time and a more well-documented API, clear semantics, and sensible defaults [16].

ElasticSearch has several advantages over Typesense, albeit having some drawbacks in specific areas. ElasticSearch offers a solid API, a distributed search engine that is easy to use and highly secure, all of which are controlled by a sizable development staff. Typesense, however, has a smaller development team and lower usage on Github [17]. The rating of the top eight open source search engines for 2020 is shown on the following graph, with red denoting quality and blue denoting advice. We can observe that MeiliSearch and Sphinx are good with relatively lower ratings, whereas ElasticSearch, Apache Solr, and Apache Lucene are all highly recommended [18]. Through table 1, Apache Lucene is one of the most recommended open source search engines. So, a case study about this open source search engine is created.

Table 1. Best Open Source Search Engines



4. A Case study

Apache Lucene is a free and open-source search engine initially written entirely in Java and supported by the Apache Software Foundation. Apache Lucene is also a widespread basis for non-research search applications. In addition, Apache Lucene is a high-performance, full-featured text search engine library. It is also cross-platform, accessible, and open source. It enables users to add search or information retrieval functionality to apps simply [19].

4.1. History of Apache Lucene

Apache Lucene was founded in 1999 by Doug Cutting, a software designer, open-source search technology supporter, and developer. Before he wrote Lucene, his other search engines were used in Apple and Excite. As the fifth search engine, Lucene was posted on the Source Forge website's headquarters and was available for download. Moreover, Apache Lucene is now managed by the Apache Software Foundation. Lucene's name has an interesting story: it was named after Doug Carter's wife's middle name and her maternal grandmother's name [20].

4.2. Significance

Why is Lucene needed? This can be attributed to two primary reasons: it can simultaneously search and query millions of files, and the application's extremely quick data-saving speed. Apps now save data fairly quickly. With this much information, a full-text search is challenging. This is since the information users want may exist in one of the network's billions of files [21]. As a result, a full-text look motor is significant. Because it can conduct lookup queries on millions of records simultaneously, people all know that information gets saved in apps quickly. In a gaming application, this is analogous to searching for a single entertainment user's records among billions of records.

The use of Lucene is common in the development of Internet search engines and local single-site searches. Examples include Apache Nutch, which provides web crawling and HTML parsing, an open source CrateDB, and DocFetcher, a multi-platform desktop search application [22]. Furthermore, many companies have adopted technology such as Twitter, Slack and Kaidee.

4.3. How Lucene operates

As seen above, it is challenging to conduct a full-text search inside a vast amount of data. So, how exactly does Lucene operate? It is important first to examine its index. Because Lucene uses inverted indexes rather than regular indexes, full-text search requests are answered quickly. In essence, an index is a data structure. Each project report includes a sample index so that readers can better understand how computer science indexes information. On the other hand, designers keep the

documents and possible placements for each word or phrase in the inverted index for every word in all documents. Additionally, the method is excellent and can simplify the search [23]. For example, if people want to find a word "This". Instead of searching the entire document in the classic index, they could use the inverted index to find the location directly.

```
Doc1 -> {"This", "is", "simple", "Lucene", "sample", "classic", "inverted", "index"}  
Doc2 -> {"Running", "Elasticsearch", "Ubuntu", "Update"}  
Doc3 -> {"RabbitMQ", "Lucene", "Kafka", "", "Spring", "Boot"}
```

Figure 2. An example of how to find work "This" in Apache Lucene

The reverse index is easy to keep up with. In other words, if users want to locate Apache, they could use the inverted index to get the answer immediately, whereas typical searches can run on the entire page, which is hard to execute in real-time settings.

```
This -> { (2, 71) }  
Lucene -> { (1, 9), (12, 87) }  
Apache -> { (12, 31) }  
Framework -> { (32, 11) }
```

Figure 3. An example of inverted index

There are four steps that Apache Lucene make when it searches the data. The first stage is to provide Apache Lucene to the documents and other data sources, and then Lucene turns that data to plain text for each document, and then the parser translates those source documents to plain text. A reverse index is then established for each term in the plain text. Finally, the index is ready for use. Lucene is a highly powerful full-text search engine using this approach. This, however, is the only aspect of Lucene implementation.

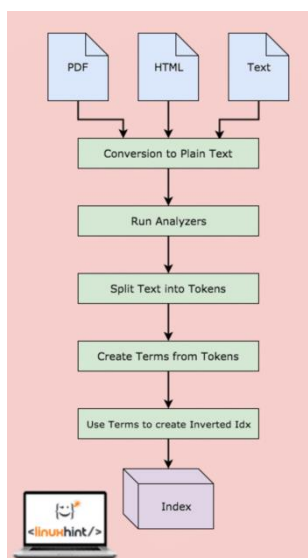


Figure 4. Four steps to search data

Two documents, as can be seen in figure 5, need to be consumed. "Jim likes playing tennis," and "Roger Federer is the greatest tennis player of all time." The information is tagged and given specific tags by the analyzer. Moreover, while keeping the relevant information, the analyzer ignores words like of, it, and the. Then, using reverse indexing, we create a database and enter the phrases for the keywords, along with their frequency and document ID. Jim only makes an appearance once in the document, as shown in the picture; hence, the frequency is listed in the table as 1. If people want to find the term "Jim," they can go directly to the document ID. The user can perform a quick search as a result of the reverse index.

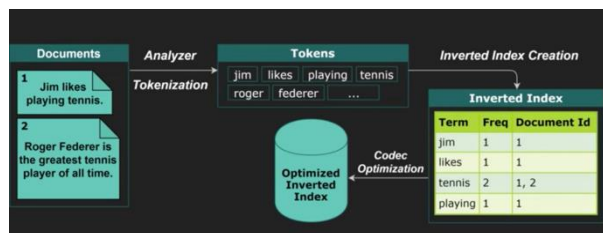


Figure 5. An example of how to get the result from reverse index

There are some basic components of Apache Lucene. The first one is the Analyzer, which is the most critical part of the indexing and search process. The parser converts plain text into tokens and terms to search for them. The next is Terms, which represents a word in the text. The last one is Documents. The data we provide to Lucene engine needs to be converted into plain text.

4.4. Main features

Apache Lucene has some main features. It can consume more than 150 GIB of memory per hour and requires only a tiny amount of RAM - only 1MB of heap. Even its incremental indexing is as fast as batch indexing. Furthermore, it has several advanced search algorithms, such as ranked search, which returns the best results first, multi-indexed search with merged results, pluggable ranking model, quick, memory efficient, and fault-tolerant suggestions [24].

4.5. Advantages and disadvantages

There are a lot of advantages of Apache Lucene. The first advantage is that it indexes quickly and efficiently. In terms of speed, no one can compete with Apache Lucene. This benefit is owing to its Java system. It simply takes a few seconds to answer a question, making it an extremely effective option for any company. It also has outstanding features that allow it to stand out from the crowd, and its efficiency makes it appealing to all business players. As the pace rises, so will the overall execution. Additionally, incremental indexing is faster than batch indexing. The second benefit is that it offers efficient and precise search engines. It ranks first in terms of algorithm, with the best result returning first. In fact, it features sophisticated query capabilities, such as proximity query, phrase query, range query, and wild card query, to provide the most accurate search. There are many search choices accessible, and we can even start a search based on the author's name, title, and other search phrases. It offers multi index search capabilities to increase the precision of the search operation and enables simultaneous search and updating to offer the best results. It also has features like highlighting, configurable face processing, and result grouping. The third benefit is that it is open source and cross-platform compatible. Apache Lucene is a free and open source search engine developed by the Apache Software Foundation. It is now available for free for any purposes, including open source development and commercial use. Because the software has no upfront costs, it is ideal for businesses who are too small to invest in software development. In addition to being written entirely in JavaScript, it may also be implemented in other computer languages. Other implemented languages, on the other hand, should be index compatible. The last benefit is similar to the second: it can find the results quickly and accurately since it uses the most efficient algorithm to support the most accurate searching and results [25].

Nonetheless, there are some disadvantages of Apache Lucene. It has a restriction when it comes to expressing complicated arguments. Apache Lucene has its version of the integral method, and when the constraints get more stringent, the Similarity class is used. However, describing complicated points, such as searches based on real matches and information, quickly reveals its limits. Furthermore, the Apache Lucene built-in does not enable clustering [26]. Apache Lucene appeared as an embedded toolkit without support for clusters in the main code. Nevertheless, there are two approaches to achieving Apache Lucene clustering: first, people may inherit and construct a Directory; second, users could people another open source search engine: Solr, people must utilize its Index Server.

4.6. The market of Apache Lucene

A new question arises: When should people use Apache Lucene? A database can be communicated with using SQL. It is the accepted language for relational database management systems, claims ANSI (American National Standards Institute). Updating data on a database and retrieving data from a database are two examples of tasks that employ SQL statements.

Therefore, in general, SQL is used for applications that require a relational data model, rely on relatively frequent record updates, or perform mathematical processing and summarization of tabular data, such as financials. Nevertheless, people would use Apache Lucene for "non-structured" text data that they wanted to search quickly. String searches for values in relational databases are typically slower than searches done on a Lucene-based system. It may be determined by how you alter the data while storing it.

Then, this study also analyzes the market of Apache Lucene. By the research there are 9785 companies use Apache Lucene, which occupies 4.65% of the market share. Moreover, among Apache Lucene customers by industry, people could find that Computer Software accounted for 33% of Apache Lucene customers and Information Technology and Services accounted for 13%. These are the most significant segments. Additionally, American businesses are the ones that use Apache Lucene the most frequently. The United States accounts for 53% of Apache Lucene users, followed by the United Kingdom with 7% and India with 6% [27].

The use of Apache Lucene is worth noticing. The Apache Lucene team keeps a list of firms that utilize Apache Lucene for their product or website, such as Twitter, a free social networking service that allows registered people to broadcast brief messages called tweets. Twitter's real-time search over tweets is powered by Apache Lucene, which handles over a billion daily requests [28]. They have altered Apache Lucene to handle the particular type of document that a 140-character tweet can be. Another application that may be used as an example is LinkedIn. LinkedIn is the internet's largest professional network. In addition, it has adapted and improved Apache Lucene for real-time and faceted search. Therefore, Apache Lucene has a great contribution to social development.

5. Conclusion

In conclusion, free, GPL, or other flexible licences, a vibrant development community, an index, keyword search, and proper ranking of the return results page are some vital characteristics of an open source search engine.

The benefits of an open source search engine are listed below. The biggest advantage of open source search engine is free, but a small part is charged. Flexibility is the second benefit of open source search engines. In order to better satisfy specific needs, developers can check that the code is working and make adjustments to the application's troublesome or dysfunctional areas. Stability represents the third benefit. Users can rely on the source code for a very long time because it is open source. Innovation is the fourth benefit. The original code can be changed by programmers to produce a better search engine. The final benefit is that these open source search engines offer beginning programmers a great chance to learn.

It is impossible to ignore the open source search engine's drawbacks. Open source engines are difficult to use because of poor settings and a complex user interface. Additionally, there are compatibility issues. Open source software also has responsibility issues. Open source code rarely provides liability or tort protection, unlike completely vendor-controlled commercial software. As a result, consumers have to follow the legal requirements. The final drawback of open source software is that it incurs unexpected costs for user training, data collection, and hardware setup. For these open source search engine advantages should be strengthened, for these disadvantages should be improved. More follow-up studies are needed. In this way our society can be better developed.

References

- [1] E.M. Clarke, E.A. Emerson, Design and synthesis of synchronization skeletons using branching time temporal logic, in: D. Kozen (Eds.), *Workshop on Logics of Programs*, Lecture Notes in Computer Science, vol. 131, Springer, Berlin, Heidelberg, 1981, pp. 52–71. DOI: <https://doi.org/10.1007/BFb0025774>
- [2] J.P. Queille, J. Sifakis, Specification and verification of concurrent systems in CESAR, in: M. Dezani-Ciancaglini and U. Montanari (Eds.), *Proceedings of the 5th International Symposium on Programming*, Lecture Notes in Computer Science, vol. 137, Springer, Berlin, Heidelberg, 1982, pp. 337–351. DOI: https://doi.org/10.1007/3-540-11494-7_22
- [3] C. Baier, J-P. Katoen, *Principles of Model Checking*, MIT Press, 2008.
- [4] M. Kwiatkowska, G. Norman, D. Parker, Stochastic model checking, in: M. Bernardo, J. Hillston (Eds.), *Proceedings of the Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation (SFM)*, Springer, Berlin, Heidelberg, 2007, pp. 220–270. DOI: https://doi.org/10.1007/978-3-540-72522-0_6
- [5] V. Forejt, M. Kwiatkowska, G. Norman, D. Parker, Automated verification techniques for probabilistic systems, in: M. Bernardo, V. Issarny (Eds.), *Proceedings of the Formal Methods for Eternal Networked Software Systems (SFM)*, Springer, Berlin, Heidelberg, 2011, pp. 53–113. DOI: https://doi.org/10.1007/978-3-642-21455-4_3
- [6] G.D. Penna, B. Intrigila, I. Melatti, E. Tronci, M.V. Zilli, Bounded probabilistic model checking with the muralpha verifier, in: A.J. Hu, A.K. Martin (Eds.), *Proceedings of the Formal Methods in Computer-Aided Design*, Springer, Berlin, Heidelberg, 2004, pp. 214–229. DOI: https://doi.org/10.1007/978-3-540-30494-4_16
- [7] E. Clarke, O. Grumberg, S. Jha, et al., Counterexample-guided abstraction refinement, in: E.A. Emerson, A.P. Sistla (Eds.), *Computer Aided Verification*, Springer, Berlin, Heidelberg, 2000, pp. 154–169. DOI: https://doi.org/10.1007/10722167_15
- [8] H. Barringer, R. Kuiper, A. Pnueli, Now you may compose temporal logic specifications, in: *Proceedings of the Sixteenth Annual ACM Symposium on the Theory of Computing (STOC)*, ACM, 1984, pp. 51–63. DOI: <https://doi.org/10.1145/800057.808665>
- [9] A. Pnueli, In transition from global to modular temporal reasoning about programs, in: K.R. Apt (Ed.), *Logics and Models of Concurrent Systems*, Springer, Berlin, Heidelberg, 1984, pp. 123–144. DOI: https://doi.org/10.1007/978-3-642-82453-1_5
- [10] B. Meyer, Applying "Design by Contract", *Computer* 25(10) (1992) 40–51. DOI: <https://doi.org/10.1109/2.161279>
- [11] S. Bensalem, M. Bogza, A. Legay, T.H. Nguyen, J. Sifakis, R. Yan, Incremental component-based construction and verification using invariants, in: *Proceedings of the Conference on Formal Methods in Computer Aided Design (FMCAD)*, IEEE Press, Piscataway, NJ, 2010, pp. 257–256.
- [12] H. Barringer, C.S. Pasareanu, D. Giannakopoulou, Proof rules for automated compositional verification through learning, in *Proc. of the 2nd International Workshop on Specification and Verification of Component Based Systems*, 2003.
- [13] M.G. Bobaru, C.S. Pasareanu, D. Giannakopoulou, Automated assume-guarantee reasoning by abstraction refinement, in: A. Gupta, S. Malik (Eds.), *Proceedings of the Computer Aided Verification*, Springer, Berlin, Heidelberg, 2008, pp. 135–148. DOI: https://doi.org/10.1007/978-3-540-70545-1_14
- [14] James Kimmons. What Is a Search Engine. September 29, 2020. Retrieved from August 12, 2022. Retrieved from <https://www.thebalancesmb.com/search-engine-2867354>
- [15] n.a.. Synopsys, Open-Source Software. 2022. Retrieved August 15. Retrived from <https://www.synopsys.com/glossary/what-is-open-source-software.html>
- [16] Chilamakuru, V. 5 open-source search engines for your website. Vishnu’s Blog. July 5, 2021. Retrieved August 16, 2022. Retrived from <https://vishnuch.tech/5-open-source-search-engines-for-your-website>
- [17] Vishnu Chilamakuru. 5 Open-Source StackShare. July 5, 2021. Retrieved August 20, 2022. Retrived from <https://vishnuch.tech/5-open-source-search-engines-for-your-website>

- [18] Dealroom. Elastic. 2021. Retrieved from August 20, 2022. Retrieved from <https://app.dealroom.co/companies/elastic>
- [19] Elasticsearch. Elastic official website. n.d. Retrieved from August 25, 2022. Retrieved from <https://www.elastic.co>
- [20] David Jeans. This Software Giant Declared War on Amazon. Will Other Open Source Companies Follow. 2021. Retrieved from August 20, 2022. Retrieved from <https://www.forbes.com/sites/davidjeans/2021/03/01/elastic-war-on-amazon-web-services/?sh=213a8b0d3dbf>
- [21] Jason Bosco. Typesense. 2016. Retrieved from August 10, 2022. Retrieved from <https://github.com/typesense/typesense>
- [22] Janani. A comprehensive guide to MeiliSearch. November 19, 2021. Retrieved August 12, 2022. Retrieved from <https://www.atatus.com/blog/a-comprehensive-guide-to-meilisearch/>
- [23] n.a..Meilisearch Documentation v0.28. n.d.. Retrieved August 19, 2022. Retrieved from https://docs.meilisearch.com/learn/advanced/known_limitations.html
- [24] Sematext. Apache solr tutorial: What is, how it works & what is it used for.Sematext. July 15, 2021. Retrieved August 12, 2022. Retrieved from <https://sematext.com/guides/solr/>
- [25] StackShare. Meilisearch vs Solr: What are the differences?. n.d.. Retrieved August 10, 2022. Retrieved from <https://stackshare.io/stackups/meilisearch-vs-solr#:~:text=Solr%20powers%20the%20search%20and,use%2C%20and%20deploy%20search%20engine.>
- [26] n.a.. What is Elasticsearch? pros, cons and features List - Courseya. <https://www.courseya.com>. n.d.. Retrieved August 20, 2022. Retrieved from <https://www.courseya.com/blog/what-is-elasticsearch-pros-cons-and-features-list/>
- [27] Luiz Lelis. Typesense search engine: an easier-to-use alternative to Elasticsearch. 2021. Retrieved from August 10, 2022. Retrieved from <https://dev.to/luizhlelis/typesense-search-engine-an-easier-to-use-alternative-to-elasticsearch-33dg>
- [28] Stackshare. Typesense. 2021. Retrieved from August 25, 2022. Retrieved from <https://stackshare.io/typesense>
- [29] Typesense. Typesense Github. n.d. Retrieved from August 25, 2022. Retrieved from <https://github.com/typesense/typesense>
- [30] LibHunt. Apache Solr vs MeiliSearch - compare differences and reviews? n.d.. Retrieved August 12, 2022. Retrieved from <https://www.libhunt.com/compare-lucene-solr-vs-MeiliSearch>
- [31] Steve Emma. 8 Best Free and Open Source Search Engines for Big Data. December 21, 2020. Retrieved August 25, 2022. Retrieved from <https://www.linuxlinks.com/searchengines/> Stackshare. Lucene. 2022. Retrieved from August 11, 2022. Retrieved from <https://stackshare.io/lucene>
- [32] Shubham Aggarwal. Introduction to Lucene. 2018. Retrieved from August 11, 2022. Retrieved from <https://linuxhint.com/introduction-to-lucene/>
- [33] Stackshare. Lucene vs Sphinx. 2022. Retrieved from August 11, 2022. Retrieved from <https://stackshare.io/stackups/lucene-vs-sphinx>
- [34] n.a.. Apache Hive vs. Apache Lucene vs. Sphinx Comparison Chart. Apache Hive vs. Apache Lucene vs. Sphinx Comparison. n.d.. Retrieved August 17, 2022. Retrieved from <https://sourceforge.net/software/compare/Apache-Hive-vs-Apache-Lucene-vs-Sphinx/>
- [35] StackShare. Lucene vs Sphinx: What are the differences?. n.d. Retrieved from August 11, 2022. Retrieved from <https://stackshare.io/stackups/lucene-vs-sphinx>
- [36] OSCOM. Open Source Search and Content Management. September 25, 2002. Retrieved August 18, 2022. Retrieved from
- [37] <https://searchtools.com/slides/oscom/OSCOM2002-08.html>
- [38] GoodWorkLabs. Three important benefits of Apache Lucene in search engine technology. 2018, March 16. Retrieved August 20, 2022. Retrieved from <https://www.goodworklabs.com/three-important-benefits-of-apache-lucene-in-search-engine-technology-goodworklabs/>
- [39] n.a.. Lucene disadvantages. n.d.. Retrieved August 20, 2022. Retrieved from <https://titanwolf.org/Network/Articles/Article?AID=b51a7ae4-c775-4428-abd2-3a9fb7e5c76d>

- [40] n.a.. Apache Lucene commands 4.65% market share in search engines. and its marketshare. n.d.. Retrieved August 20, 2022. Retrived from <https://enlyft.com/tech/products/apache-lucene>
- [41] Quora. Which major companies are using lucene?. n.d.. Retrieved August 20, 2022. Retrived from <https://www.quora.com/Which-major-companies-are-using-Lucene>