A Comparison of Various Open-source Search Engines

Neelansh Visen

2021 January 01

Abstract

Search engine providers concern themselves primarily with crawling, indexing, storing, and displaying web pages to internet users. The advent of the information age has manufactured individual user data into a valuable, highly sought-after commodity that if exploited runs risk of exposing personal information to the masses. These risks, in conjunction with a lack-of transparency from big-technology companies, has bolstered the demand for alternative, open-source search engines. The following report seeks outline the history, principles, functions, and features of six of these open-source search engine projects.

Contents

1 Introduction

As the sea of information available on the internet rises, so too does the importance for users to be able to traverse, identify, and retrieve specific and accurate search results. The introduction of web crawling, indexing, and page ranking pages on the world wide web has dramatically improved the accessibility, efficiency, and quality of the average user experience on the internet. Dating back to the hand-maintained servers of Tim Berners-Lee, the sudden growth of the internet during the 1990s paved the way for companies like Yahoo! and Google to cement their imprints on the search engine market.

As any internet user knows, a search on the web is made by opening a web browser and inputting a search query (a word or phrase of interest) into a text box or bar. At the core of this process lies the search engine, a software system concerned with crawling, indexing, sorting, and displaying web pages, images, videos, articles, and other types of data on the internet. Generally speaking, search engines are designed to provide results based on several factors, including relevancy, popularity, and in many cases user data (a process leading to what is called a filter bubble) [1]. Generally an organic process, there are times when search engines will deliberately filter in or out search results based on socio-political and commercial factors. Due to prevalence of the internet in the modern age, these implicit biases can have severe consequences when it comes to macroscopic behavioral trends like consumer preferences and voter intentions.

With billions of dollars in advertising revenue at stake, the algorithms, methods, and practices underlying the search engines of tech-giants such as Google, Microsoft, and Baidu are hidden behind closed doors. The inherent secrecy of these companies and their technologies has raised concerns when it comes to privacy, data exploitation, and user-profiling at-large. Although these companies claim to use collected data for the benefit of the user, reports of hacks, breaches, and government subpoenas have increased the demand for a viable alternative to big-tech [2].

1.1 What is an Open-Source Search Engine?

One of the most prevalent remedies to these privacy concerns has come in the form of open-source search engines. Like other open-source software, open-source search engines share their source code publicly under a software license granting parties the ability to study, modify, and share exact or modified versions of the software for any purpose, generally at no cost. Open-source search engines provide similar functions to their proprietary alternatives in their abilities to browse the web while placing a greater emphasis on user transparency, actively maintained software, and flexibility in customization.

1.2 Why Open-Source?

Open-source is preferred over proprietary software for a number of reasons including security, control, training, stability, and community [3]. In terms of security, the money involved in proprietary software may incentivize questionable or unethical business practices. In addition, the aforementioned privacy concerns and compromises in security are often enough to dissuade users. As for control, many users and developers prefer the flexibility provided to them by open-source search engines. Open-source code scripts can be examined thoroughly from ground-zero, and various settings and functions can be requested or designed from scratch. Furthermore, open-source projects permit users to use the software for whatever purpose they wish, unlike proprietary software that may restrict the domain of uses for a particular piece of software. When it comes to training, the accessibility of well-written, publicly maintained and deployed open-source software may also encourage interested users to hone their skills, leading to improvements in current projects and development of derivative projects. This point leads directly to the importance of stability to open-source technology. The public distribution of software means that critical aspects of a project will not disappear suddenly from the public domain. By design, open-source software, and as a result open-source search engines, promotes open collaboration and sharing amongst users, bolstering a community dedicated to providing value to the masses without financial compensation.

2 YaCy

2.1 Overview

YaCy is a freely distributed search engine founded on the principles of peer-to-peer connections. Unlike most other search engines, YaCy does not run search queries through a central server. Instead, YaCy users download the available software onto their computers where it conducts its own personalized search indexes and rankings. Computers that have installed and run YaCy software are referred to as YaCy peers. Each peer is equal, and the connections between peers serve as an alternative to a common central server. Mapped out as a "freeworld" network, YaCy is the largest open source search engine comprising of hundreds of users on a global scale.

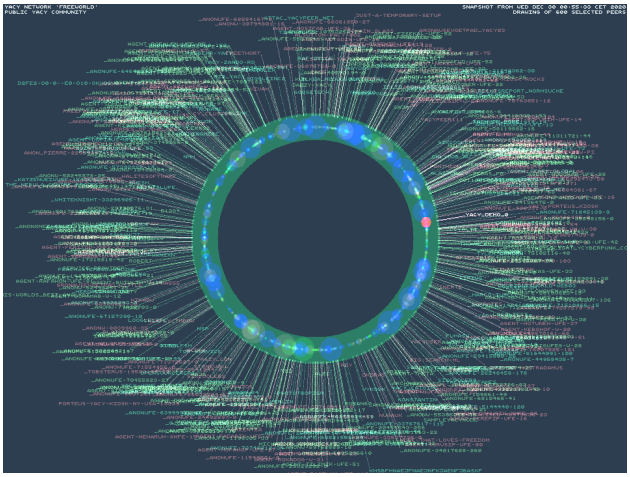

Figure 1: The YaCy Global Network

2.2 History

Authored by Michael Christen, YaCy was first released in 2003. Speaking on his motivations to produce the open-source project, Christen commented, "Most of what we do on the Internet involves search. It's the vital link between us and the information we're looking for. For such an essential function, we cannot rely on a few large companies, and compromise our privacy in the process," [4]. Like other open-source search engines, the rationale behind YaCy was driven by a lack of viable decentralized services without a centralized point-of-

control. Beyond addressing privacy and security concerns, a lack of centralization meant the elimination of a single point of failure for the technology itself as well as the, "potential for faster responses through load-balancing" [4].

2.3 Methods and Components

At its core, YaCy is a distributed web search engine, meaning a search engine with no central server or singular point of control. Instead, a network of YaCy peers (each computer with YaCy installed on it) sustains a shared reverse-word index for every crawled web page. Broadly, this shared index is a database of matching uniform resource locators (URLs) that has been ordered based on the likelihood of search. For YaCy, the database of indices is sharded or horizontally partitioned amongst each YaCy peer in a distributed hash table or DHT. Every time a peer indexes a fresh set of web pages, the DHT is updated. YaCy's DHT does not store the entire URL reference for every matching page however, but rather metadata related to crawl time, language, and file type, things that might be of use to the searching user. In order to increase lookup speed and availability, these pieces of information are stored locally on the YaCy peers' machine, and are replicated onto several other YaCy peers [5]. This data is kept as an SQL database using an AVL or self-balancing binary search tree that permits lookup, insert, and delete operations in logarithmic time. Every word in the YaCy's DHT index contains two hashes, one for every particular word containing a hash of the URL and one for every reference of storage. YaCy separately stores these hashes separately because URLs matching multiple search terms only have to be fetched from a peer once for any given search. Additionally, because the number of URL references is uncontrollably large, YaCy partitions word entries across multiple YaCy peers. While this additional measure balances the load between machines, it adds an additional layer of complexity to the search process.

This complexity is amplified since YaCy does not use a central server to assemble and sort results. That job is left to the local YaCy web application, which collects results from a local database and from remote YaCy peers. When a user performs a search on YaCy, a query is started on that user's local database. Each search result is cached locally, so as

to limit taxation on the network while increasing the accuracy and reliability of subsequent searches by that user. Results are kept together in a Reverse Word Index Queue, where a ranking algorithm akin to Google's PageRank is applied. Similar to browser cookies, YaCy observes a user's search actions and uses them to refine future search outputs. Content fetching, though time-consuming, is conducted by YaCy to filter out spam. By fetching the contents of each page, search terms can be compared directly to the contents of a web page.

2.4 Features

Written in the programming language Java, YaCy is supported on Windows, Mac, and Linux operating systems. As for the search interface, YaCy offers users the ability to specify results based on file type, domain, and authorship. If desired, users are able to crawl and index the web themselves, and can decide whether a search is to be conducted locally or as a global DHT-based search. YaCy also classifies peer status based on contact and connection to networks and servers as follows. Status virgin means the peer is not in contact with the network. In other words, the user is considered offline and is limited to searching the local index only. Status junior indicates a peer that is connected to the YaCy network but cannot be accessed by other connected peers. This status can come about if a particular firewall or router configuration is producing interference. Status senior is designated if a connected peer is accessible by other YaCy peers, permitting searches on the local and global index while becoming an access point for index distribution. The highest status, principal, is a senior peer that has, "uploaded an additional peer-list to a server...[supporting] other peers to get in contact with the existing YaCy network to perform a global search.

YaCy prioritizes user safety by preventing the indexing of all password and cookie protected pages on the web. Furthermore, any pages loaded using a GET or POST request by default are barred from indexing. As for customization, "the administrative interface allows you to configure the YaCy client in several other modes, including intranet indexing and deriving as a single-site search portal" [5]. User accounts themselves can also be password-protected (as is the case for certain company or organization intranets).

3 Apache Lucene

3.1 Overview

Apache Lucene (or simply Lucene) is an established open-source search engine library that operates via search and inverted full-text index. Supported by the Apache Software Foundation and licensed under the corporation's eponymous license, Lucene is a technology suited for nearly any application that relies on full-text search, particularly in cross-platform environments. Unlike searches based on metadata, full-text searches examine every word in a stored document while finding a match to the search criteria. This method leads to what is known as the precision-recall trade-off. In pattern recognition, information retrieval, and classification, precision measures the quality of returned results, whereas recall measures the quantity of relevant results returned by a search. Generally, full-text search applications like Lucene allow users the flexibility to tune down precision in favour for better recall.

Figure 2: Doug Cutting

3.2 History

A by-product of Doug Cuting's desire to learn Java, Lucene was the software designer's fifth search engine. Named after Cuttings' wife's middle name, Lucene was first

made available via SourceForge in April of 2000. 17 months later, Lucene was donated to the Apache Software foundation's Jakarta Project, an initiative designed to house open-source Java products. In 2002, Lucene had seen nearly 400 commits from 13 different programmers. By 2005, Lucene had become a top-level Apache project, later overlooking sub-projects such as Lucene.NET, Mahout, and Tika.

3.3 Methods and Components

It is important to not that Lucene is not a web crawler (see: Apache Nutch), nor is it an application or library for link analysis algorithms. Rather, Lucene is a library that enables full-text based search. To do this, Lucene uses what is known as inverted indexing; essentially a method of managing an index over a dynamic collection of documents. In particular, Lucene indexes for two types of terms: those generated from non-text fields and those generated from text-fields. Non-text field terms are given a field name and field value, whereas text field terms are given field name and token. Lucene provides a mapping from these terms to documents via inverted indexing, reversing the typical mapping of a document to the terms it contains, thereby providing a mechanism for ranking search results based on relevancy. The greater the number of present search terms in the document, the more likely the document is to be relevant to the searching user.

Whereas conceptually indexing and search is conducted over documents, in reality all indexing and search is applied over fields. Simply, a document is a collection of several fields containing three parts: a name, type, and value that upon search are used to narrow the breadth of search to particular fields. For example, when indexing a medical citation entry with Lucene, designated fields may include the name(s) of the author(s), the name of the journal under which the citation was publishes, the title of the article, the date of publication, and the text found in the article's abstract [6]. Because each field is given a different name, users can specify the type of field being searched upon (e.g. a range of dates for journal publications).

As for indexing documents with Lucene, the process begins by creating a document containing fields that need to be indexed and later stored, followed by adding said document

to the index. Lucene uses directories to provide a seamless interface akin to an operating system's file organization system. Within directories lie sub-indices called segments that allow Lucene to update and delete documents from the index in an efficient manner. Only when a document has been indexed and stored in a retrievable manner is a user able to search. Using a query parser, the query (i.e. a string of characters) is converted into an object in Java which is handed to Lucene's index searcher before ultimately returning a list of ranked web pages or "hits" based on relevancy to the original search query. Because Lucene is only able to index strings of text, an added process of preparing and adding text to Lucene may be required. Essentially, it is up to the user to convert a particular file format to something that Lucene can retrieve through inverted indexing.

3.4 Features

Lucene maintains a custom language in the form of Boolean and other logical operators. These operators assist users in specifying or broadening any given search query. One interesting feature Lucene provides is the ability to identify a specific term or clause of a query as more important than the others, a feature known as "boosting". Other notable search functions Lucene provides include keyword matching, wildcard matching (searching for words or phrases that begin or end with a specific term), and proximity matching (searching for multiple words in a query within a specified distance of one another) [7].

As for key features, Lucene is scalable and performs high performing indexing. The software is able to adapt to increasingly large amounts of data with ease. Given a user's hardware, Lucene can process up-to 150 gigabytes of data per hour using a tiny amount of RAM (1 megabyte). As for search efficiency, Lucene's ranked searching is conducted using powerful, accurate, and efficient search algorithms. Because Lucene is written in Java, the software is able to run across multiple operating systems, including Android and iOS. Furthermore, Lucene's relatively straightforward API makes it easier for developers to produce work with more autonomy, greater scope, and simpler integration, adaptation, customization, and personalization. On top of these features, Lucene is free to download and is currently used by mega-corporations such as Twitter, LinkedIn, and Comcast.

4 Apache Nutch

4.1 Overview

Apache Nutch (or simply Nutch), is a highly extendable and scalable open source web crawler project. Like Lucene, Nutch is a product released under the Apache Software Foundation. Composed of two branches called Nutch 1.x and Nutch 2.x, Apache Nutch essentially allows users to create their own search engines on their machines. Nutch installations typically work on one of three scales: the local file system, the intranet, or the entire web. Built on top of Lucene, Nutch implements a "map reduce" distributed model, meaning that it processes and generates big datasets using parallel, distributed algorithms on computer clusters. Whereas Lucene is useful if a web crawler application is not needed, Nutch is a better fit for web pages where direct access to underlying data is unavailable.

4.2 History

Nutch was authored by Lucene creator Doug Cutting and Hadoop co-founder Mike Cafarella and hosted at SourceForge. Backed at first by a private non-profit, Nutch gained the Apache license in January 2005, two years after its original release. Originally housed under the Apache incubator, Nutch eventually became a sub-project of Cutting's previous project Lucene, graduating eventually to "top-level" status in April 2010. In 2012, Nutch launched its 2.x branch focused on large scale web crawling built on storage abstraction [8]. As of July 2020, Apache has launched Nutch versions 1.17 and 2.4.

4.3 Methods and Components

As a web crawler, Nutch builds a categorized index of web pages for search engines in six steps. First comes injection, where Nutch reads a URL list from a given seed text, "injecting" them into the crawl database to initiate the crawling process. The crawl database contains information on every URL known to Nutch, such as if and when the URL was fetched by the software. The second step involves generating a fetch list from the crawl database. The list contains URLs that were previously injected into the crawl database, and

is eventually placed into a segmented directory. In the next step, content is fetched from the URLs stored in the fetch list. During this process, Nutch's web crawler systematically scans pages from the web, eventually writing the information back into a segmented directory. The information from this directory is eventually parsed and cleaned. By cleaning, Nutch removes any unnecessary tags (e.g. HTML tags) from entries depending on various parameters. This step converts stored objects into a format known as "ParsedData" that includes text data as well as any available out-links [8]. Once the content has been organized and stored, the crawl database is updated with information from the fetcher and parser. This information may include hyperlinks from URLs and updates page rank and web link details. Eventually, the cleaned, organized, parsed data is placed into an index connected to an interface searchable by users. This process is repeated for every update to the Nutch directory.

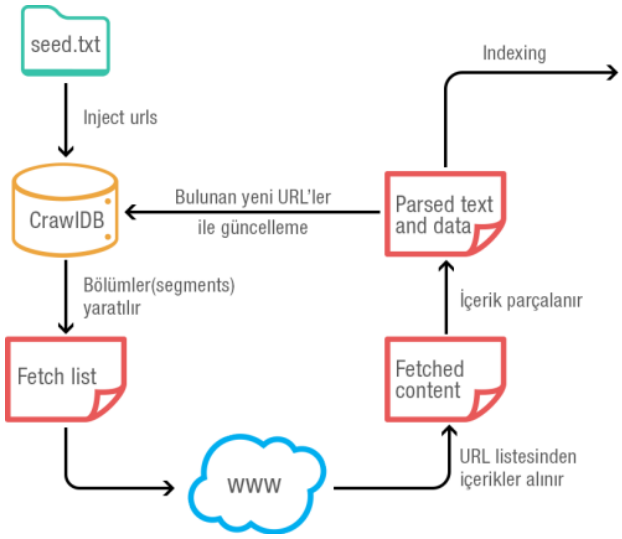

Figure 3: The Nutch Crawling Process

4.4 Features

One of the advantages Nutch has over other web crawlers comes with its speed. At over 6000 URLs per minute, Nutch's parsing speed fluctuates based on crawler parameters, the types of sites being crawled, as well as a user's bandwidth limitations. Nutch is highly scalable and flexible, and like Lucene is written in Java. In particular, Nutch supports MySQL and Mongo databases, as well as searchable interfaces such as Solr and Elasticsearch.

5 The Lemur Project

5.1 Overview

The Lemur Project is a collaborative effort between the University of Massachusetts, Amherst, and Carnegie Mellon University [9]. The purpose of the Lemur Project is to develop open-source search engines, tools for text analysis, browser extensions, and other data retrieval resources intended to enhance the growth and development of research pertaining to information retrieval algorithms and text mining software.

5.2 History

Founded in 2000, the Lemur Project was a joint venture launched by the Center for Intelligent Information Retrieval at Amherst and the Language Technologies Institute at CMU. Over the years, students and faculty at both universities have contributed to several open-source endeavors under the Lemur Project, beginning with the Lemur Toolkit, "a collection of software tools and search engines designed to support research on using statistical language models for information retrieval tasks" [9]. The Lemur Project would later include the search engine Indri, the Lemur Query Log Bar, and the datasets ClueWeb09 and ClueWeb12. Generally, major software changes are released in June and December, with minor changes released as necessary. Whereas the majority of these changes come from people associated with Amherst and CMU, "the project welcomes software, documentation, data, and other contributions from the broader Lemur community" [9].

5.3 Methods and Components

The Lemur Project's Indri search engine is designed to address four primary goals. The first is to support complex queries involving evidence combination and the ability to specify a wide variety of constraints involving proximity, syntax, extracted entities, and document structure. The second is to create a retrieval model with superior effectiveness across a range of query and document types including web, cross-lingual, and ad-hoc. The third is to create an interface supporting retrieval at different levels of granularity. These

levels range from sentences passages and XML fields to documents and multi-documents. The fourth goal of the Indri search engine is to have a system architecture that supports massive databases, multiple databases, optimized query execution, fast indexing, concurrent indexing, and querying. Beyond these goals, "Indri is a search engine that provides state-of-the-art search and a rich structured query language for text collections of up to 50 million documents (single machine) or 500 million documents (distributed search)" [10].

Indri's retrieval model combines two approaches to retrieval: a language modelling approach [11] and an inference network approach [12]. Documents are ranked according to the belief that the information need "I" is met giving document "D" and hyper-parameters alpha "a" and beta "b" as evidence. Documents are represented as multi-sets of binary feature vectors, unlike the sequences of tokens and terms present in typical language modelling frameworks. This representation allows for more interesting text features to be explored such as specific phrases, absences of terms, or words that are specifically capitalized. Because search queries are composed of a series of terms, phrases and complex entities, a document is considered relevant only when it contains the same concepts as those listed in the query. Indri uses inference networks to combine sources of document relevance, a process mapped below in figure 4. In this image, document features are represented by "r" nodes and are presented as evidence to concept nodes depicted by "q".

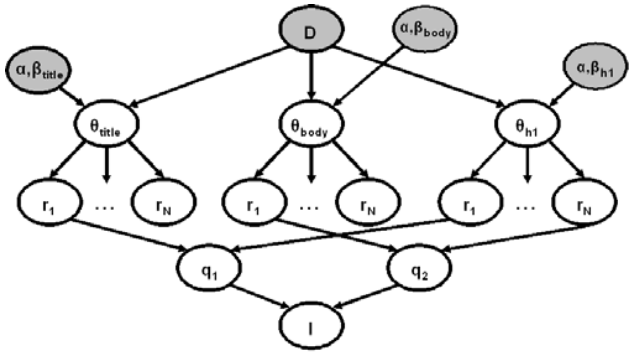

Figure 4: A Model of an Inference Network

5.4 Features

The Indri search engine contains a powerful query interface, that supports popular structures query operators from InQuery, suffix-based wildcard term matching, field retrieval, and passage retrieval. As for flexible indexing and document backing, Indri supports UTF-8 encoded text, language independent tokenization of UTD-8 encoded documents, PDF, HTML, XML, and TREC parsing, Microsoft Word and PowerPoint parsing (on Windows only), text annotations, and document metadata.

When it comes to package versatility, Indri is open-source, with a flexible BSD-inspired license. This family of licenses imposes minimal restrictions on the distribution and use of software licensed under this name, and differentiates itself from copyleft licenses in that derivative works are not restricted by the same licensing agreement. Indri also includes both command line tools and a Java user interface with a simple API that can be used with Java, PHP, or C++. In terms of scalability and efficiency, Indri has best-in-class ad-hoc retrieval performance. furthermore, Indri can be used on a cluster of machines for faster indexing and retrieval, scaling terabytes sized collections with relative ease.

6 Searx

6.1 Overview

Searx is a free, metasearch engine available under the GNU Affero General Public License version 3, with the aim of protecting the privacy of internet users. Essentially, metasearch engines (also known as a search aggregator), are an online information retrieval tool that uses other search engine queries to produce its own results. Like any other search engine, a metasearch engine takes an input in the form of a search query and gathers, ranks, and presents web data to the users the same way any other engine would. The difference of course is a matter of where the search data is coming from.

6.2 History

In March of 1995, Daniel Dreilinger out of Colorado State University unveiled his personal venture SearchSavvy. At the time, Dreilinger's motivations were simply the fact that no major search engine at the time could ever contemplate crawling the entire web on its own. As the original metasearch engine, his project aggregated more than just web content, but also specialized databases like Shareware.com and IMDB. It gave users with a "search plan" to execute their query against the most relevant subset of engines and databases for their query. Like most metasearch engines that followed, SearchSavvy's goal was to optimize two conflicting tasks: minimizing resource consumption while maximizing search quality. In the years following the release of SearchSavvy, a consolidation of the search engine market led to the release of several new metasearch engines. In January 2014, Adam Tauber launched the first iteration of his search engine, Searx.

6.3 Methods and Components

Searx provide users an increased breadth of search for data from the web. Metasearch engines in general have the advantage of running a search query on multiple engines concurrently, allowing users utilize lesser-known engines for their searches. The application of several engines to a single search helps users discover websites that may slip through the

cracks of any one search engine. This process saves users time and exertion, making search engines an excellent choice for overviews and quick answers. Metasearch engines are not completely excused of their drawbacks however. Searx is not capable of parsing search query forms or fully translate search query syntax on its own. Rather, it simply works as aggregator or results derived from more advanced engines such as Google and Bing. Because Searx does not have a general search API, an adapter has to be built between Searx and every search engine it derives its results from.

Unlike major search engines, which display search results as tracked redirect links, every search result found through Searx is given via a direct link to the respective website. These results are found after a user inputs a query into a search bar. Behind the scenes, queries are made using a POST request on every search engine except for Google, which uses a GET request. POST and GET are two communication methods in HTTP that are carried out between users and internet servers. In addition to regular search, Searx supports search within specific domains including: files, images, IT, maps, music, news, social media, and videos. In terms of user functionality, Searx contains various operators that can be applied prior to entering a search query. These operators include category specification, engine specification, language specification, safe-search, and time-range. Searx also allows users to specify whatever engines they want conducting any particular search through a preferences interface that is saved locally to a user's browser. As of 2020, Searx supports roughly 70 search engines, and maintains, "easy integration with any search engine" [13].

6.4 Features

6.4.1 Free Software

Released as "free" software under the GNU Affero Public License, Searx is, and will forever be available to the public as free software. Free software is any software that is distributed under terms that allow users to run the software for any reason, granting users the freedom to study, change, and distribute the original and any adapted versions of the software. Free software is designed as a matter of liberty, based on a philosophy focused on liberating users in cyberspace, granting everyone the ability to share knowledge and learn

from others. Led by programmer Richard Stallman, the Free Software Movement is founded on four fundamental freedoms. The first freedom is the freedom to study how the program works and change it so it does your computing as you wish. Essentially, if your right to modify a piece of software is limited, then that software is not considered to be free. The second freedom is the freedom to redistribute copies with the purpose of helping others. This freedom grants users the ability to not have to ask or pay for permission to redistribute an original piece of software. The third freedom states the freedom to distribute free copies of modified free software to others. In conjunction with the second freedom, these rules hope to give the entire developer community a chance to benefit from free software without limitation. Stallman's fourth rule, dubbed rule 0, grants users the freedom to run a piece of software for any purpose. That is, no one should be able to impose their purposes, instructions or guidelines onto any user of free software [14].

6.4.2 Privacy

Searx places a significant emphasis on protecting user privacy. A prevalent issue for many internet users, major search engines are known to use tracking cookies, log specific search queries (browser history), log the IP addresses of various devices, and collect personal data via user profiling. A lack of rigid regulation has emboldened firms to abuse the data they collect, paving the way for open-source projects like Searx to come up with viable alternatives. Searx looks to protect user data in three ways. First, Searx removes all private data from search requests sent to search services. In other words, search history is not sent to Google, Bing, or any other search engine company. Second, Searx does not forward anything from third-party services (i.e. no advertising). Third, Searx hides private data from reaching results pages. According to their documentation, "removing private data means not sending cookies to external search engines and generating a random browser profile for every request" [13]. While these measures prevent third-parties from preying on user data, the trade-off comes in the form of personalized advertisements, relevant results, and general convenience.

7 Sphinx

7.1 Overview

Sphinx is a full-text, free and open search server focused on performance, relevance in search quality, and simplicity in integration.Written in C++, Sphinx runs on Linux, Windows, MacOS, Solaris, and other lesser-known operating systems. Sphinx focuses on batch indexing and searching data stored in SQL and files in general. Text processing enables users to customize Sphinx in accordance to their desires, and a basix API allows for search in just 3 lines of code [15]. Sphinx clusters scale over ten billion documents and one hundred million search queries every day. Sphinx software backs companies including Craigslist, Living Social, MetaCafe, and Groupon, and is licensed under the GNU General Public License version 2.

7.2 History

Given the name based on an acronym for **S**QL **Ph**rase **In**de**x**, Sphinx was created by Andrew Aksyonoff in 2001. According to the founder, "Sphinx development was started...because I didn't manage to fund an acceptable search solution (for a database driven Web Site)" [16]. In Sphinx, Aksyonoff looked to improve upon three issues plaguing database search engines at the time. First, Aksyonoff looked to improve search quality. In particular, the statistical ranking methods which worked on large collections of small documents. Second, Aksyonoff sought to increase search speed, especially if search queries contained stop words such as "the", "at", "which", and "on". Finally, Aksyonoff desired a software that would reduce disk and CPU requirements while indexing the web.

7.3 Methods and Components

These types of queries include words and phrases as well as multiple forms of a word or phrase. To support full-text queries, full-text indexes are implemented on character-based data columns in a table (See Figure 5 below).

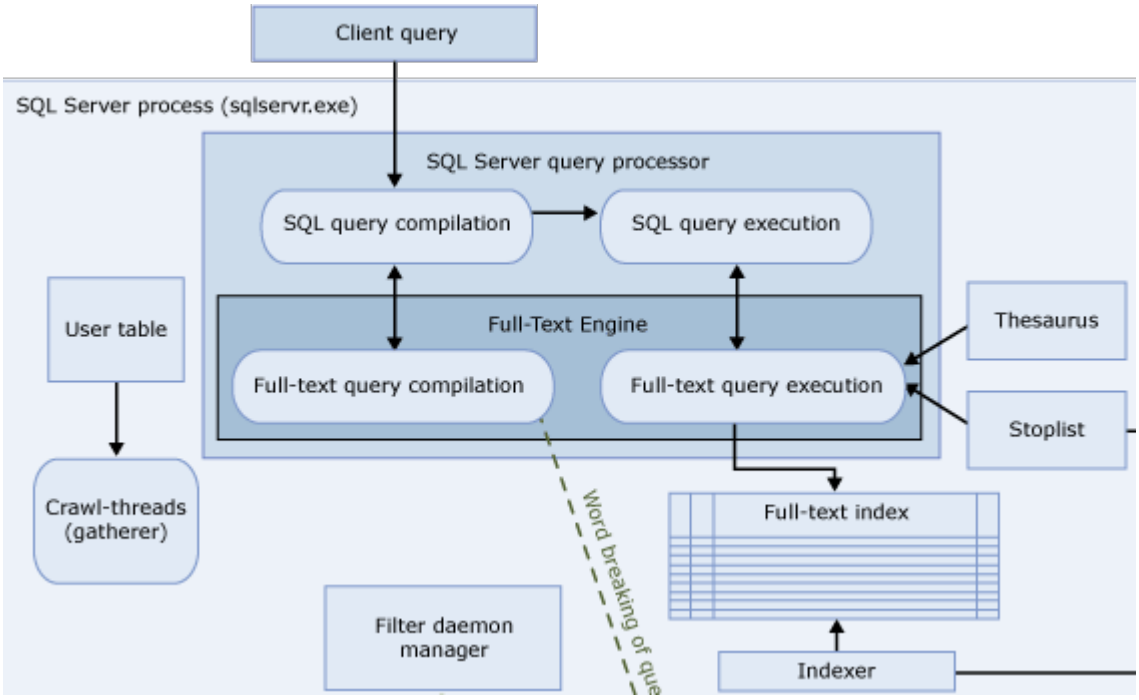

Figure 5: A Full-Text Index Process

Whereas traditional search only looks through parts of text file such as a title, abstract, and references, full-text search examines all of the data. Traditional search engines tend to provide users with a broad set of results that require users to continue to search and refine. Meanwhile, full-text search engines will match every term the user is interested in, thus providing a narrower set of results. Full-text indices contain a list of text data and a respective position within a document. This method of indexing has several advantages over traditional indexing and search methods that do not employ full-text search. Searches tend to be quicker and more relevant due to the lower amount of results being returned to the user. Furthermore, full-text search performs well on huge databases and is able to parse through common words more effectively than traditional indexing methods. Due to the size of full-text indices however, one potential drawback of the method is that it can take up quite a bit of space on its machine.

Sphinx's search workflow is composed of four main components: a client program or application, a database, an indexer, and a daemon or background process called searchd, outlined in the figure below (See Figure 6).

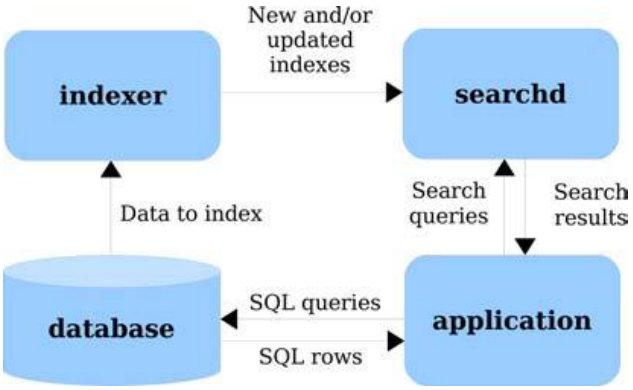

Figure 6: The Sphinx Workflow

Generally speaking, the application receives a search query from the user and sends it to the searchd program to display the returned results. The data stored in the database (usually MySQL or some other SQL server) is queried by the indexer, which in-turn generates a full-text index based on the data. The searchd program communicates with the application and uses the index generated by the indexer to parse through search results (the same filtering, grouping, and sorting methods that all search engines apply to their results).

7.4 Features

There are may reasons why Sphinx is ideal for full-text searching. Sphinx uses batch and real-time full-text indices, supports non-text attributes, SQL and non-SQL storage indexing, and easy integration. Additionally, Sphinx uses advanced full-text searching syntax and relevance rankings, rich, database-like querying features, flexible text processing, and distributed searching. As for performance and scalability, Sphinx can index up to 15 MB per second per CPU core. On a standard 2-core desktop machine with 2 GB of RAM, Sphinx is able to run over 500 queries per second against one million documents, with the biggest Sphinx clusters indexing over 25 billion documents.

8 Conclusion

Living in a time where big technology companies dominate our everyday lives, it can be difficult to maintain any semblance of privacy online. Given the permanence of the internet, protecting user information is and will continue to be of the utmost importance. Desensitized to personalized advertisements and unusually accurate recommendations, there are moments online where it starts to feel like Google and Bing are living inside of your home. Although the sophistication of modern software has made certain aspects of the internet much more accessible and user-friendly, privacy is certainly not one of them.

Open-source search engine projects seek to liberate the average internet user, granting them safe access to the near infinite possibilities of the internet without repercussion. This report summarizes key findings and observations of a presentation completed on December 1st, 2020 for APM306 at the University of Toronto. Researched by myself, Hana Uddin, Isabella Vari, Gagandeep Singh, Stanley Zhang, and Temilade Adeleye, this report presents an overview, history, methods, components, and features of six different open-source search engines available for free online.

9 References

[1] E. Pariser, The Filter Bubble: What The Internet Is Hiding From You. London: Penguin Books, 2012.

[2] R. McMillan and R. Knutson, "Yahoo Triples Estimate of Breached Accounts to 3 Billion," wsj, 04-Oct-2017. [Online]. Available: https://www.wsj.com/ articles/yahoo-triples-estimate-of-breached-accounts-to-3-billion-1507062804.

[3] The Open Organization, "What is open source?," opensource, 2020. [Online]. Available: https://opensource.com/resources/what-open-source.

[4] I. Thomson, "YaCy takes on Google with open source search engine," theregister, 2011. [Online]. Available: https://www.theregister.com/2011/11/29/yacy_google_open_source_engine/.

[5] N. Willis, "YaCy: A peer-to-peer search engine," lwn, 2011. [Online]. Available: https://lwn.net/Articles/469972/.

[6] R. Karim, "Searching and Indexing With Apache Lucene - DZone Database," dzone, 2017. [Online]. Available: https://dzone.com/articles/apache-lucene-a-high-performance-and-full-featured#: :text=Apache Lucene is a high,in a cross-platform environment.

[7] K. Tan, "Lucene Query Syntax," lucenetutorial, 2020. [Online]. Available: http://www.lucenetutorial.com/lucene-query-syntax.html.

[8] Apache Nutch, "Highly extensible, highly scalable Web crawler," nutch, 2020. [Online]. Available: http://nutch.apache.org/#:̃text=23 March 2009 - Apache Nutch, The release is available here.

[9] The Lemur Project, "About The Lemur Project", lemurproject, 2018. [Online]. Available: https://www.lemurproject.org/about.php.

[10] The Lemur Project, "Indri," lemurproject, lemurproject, 2020. [Online]. Available: https://www.lemurproject.org/indri.php.

[11] J. M. Ponte and W. B. Croft, "A language modeling approach to information retrieval,"

acm, 1998. [Online]. Available: https://dl.acm.org/doi/10.1145/290941.291008.

[12] H. Turtle and W. B. Croft, "Evaluation of an inference network-based retrieval model," acm, 1991. [Online]. Available: https://dl.acm.org/doi/10.1145/125187.125188.

[13] A. Tauber and N. Ványi, "Welcome to Searx," searx, 2020. [Online]. Available: https://searx.github.io/searx/.

[14] R. Stallman, "What is free software?," gnu, 2019. [Online]. Available: https://www.gnu.org/philosophy/free-sw.en.html. [Accessed: 01-Jan-2021].

[15] Sphinx Technologies Inc., "About," sphinx, 2021. [Online]. Available: http://sphinxsearch.com/about/sphinx/.

[16] A. Aksyonoff, "Sphinx 2.2.11-release reference manual," sphinx, 2016. [Online]. Available: http://sphinxsearch.com/docs/current.html.