DOI https://doi.org/10.15407/csc.2021.05-06.055
UDC 004.4

H.V. KHODIAKOVA, PhD (Educ.), Associate Professor at the Department of Computer Science of the V. O. Sukhomlynsky Mykolaiv National University, Shneerson ave., 11, Mykolaiv, 54001, Ukraine,
khodiakovagalina@gmail.com

N.V. KHODIAKOVA, Senior Software Developer at Ray Sono AG, Bruderhofstrabe 3, Munich, 81371, Germany,
nathalie.mk.ua@gmail.com

V.A. POZDEEV, Doctor of Physical and Mathematical Sciences, Professor, of the V. O. Sukhomlynsky Mykolaiv National University, str. Nikolskaya, 24, Nikolaev, 54030, Ukraine,
valer.al.pozdeev@gmail.com

FULL-TEXT SEARCH SETUP ON A WEBSITE

A method for implementing full-text search using Elasticsearch is described. The advantage of the new approach is the asynchronous sending of the page content and its address to a specific service responsible for communication with Elasticsearch. This allows you not to block the normal work with the CMS and not depend on the availability of the indexing service. The approach described in the article is flexible and adaptable for various website architectures. Asynchronous processing of indexing requests ensures high query execution speed and system fault tolerance.

The article discusses various approaches to implementing full-text search on a website, their advantages and disadvantages. Based on the analysis, a more flexible and universal approach to the implementation of a full-text search system has been developed. A solution is proposed with a step-by-step implementation and setup of advanced full-text search using Elasticsearch.

***Keywords**: Full-text search, algorithm, natural language processing, search robot, website, text indexing, search engine.*

Introduction

Full-text search is the search for words or phrases in textual data. For websites, the term "full-text search" is used when a user needs to find a word or phrase in the whole text available on the website. The developer cannot always successfully fulfil this task, because it is quite multifaceted [1]. At the same time, the task of full-text search is now very popular and relevant. Constant on-going search for new methods of solving this problem shows that. There are many publications about this problem on the Internet. The articles [1–4], and other ones are dedicated to the general concepts and examples of specific solutions for full-text data search. We did not find any researches dedicated to this problem in the printed editions.

Problem Setting

Modern requirements for website development include a full-text search function. Approaches of different complexity and productivity are used in the implementation of the search function. There is also a sequence of related tasks: selecting the option of indexing text, sending text for indexing, selecting texts for indexing directly from the CMS (Content Management System) database, selecting a search engine and others.

The purpose of this study is to analyse existing solutions for full-text search on the website, to

choose the most universal, stable and scalable approach, as well as to find a solution for customizing full-text search with Elasticsearch.

An Overview of Approaches to Implementing a Full-text Search on a Website

An independent implementation of the search algorithm can be considered as the simplest and most primitive solution from the developer's point of view. If the website is presented entirely as a source string, and the user's query is shown as the search data, then the task is to find the substring in the string. This is one of the basic problems of algorithmization, which is solved by fairly common algorithms. Article [2] mentions a whole class of algorithms of the substring search in a string.

Simplicity of implementation is the main advantage of this approach, but its disadvantages are also obvious:

- As the website grows, the output string where the search takes place grows.
- Repeated search queries for the same words take as long time as the original query. Lack of optimization will lead to an increasing load on the site with agrowing number of users.
- The search assumes the exact match of the search query to the given text. Even if a number of characters differ (Hamming distance search), this search will ignore more complex typos or synonyms in the search query.
- By presenting the whole site as a source line for search, we do not distinguish between the logical parts of web pages — the title, the body of the page and the footer. If a user query is found, for example, in the site menu, and the menu is on each page, then absolutely all pages will meet the search criteria. However, when searching for words, the user is more interested in the content of the page than the content of the menu.

So, there are a lot of problems that appear with this approach. The search algorithm used is quickly becoming more complex, and at some point there is a need to use another solution.

Existing search engines can be used to increase productivity and reduce search time. These systems implement already known algorithms of the substring search in a string, but for this purpose they use pre-processing, pre-indexing the text [5].

The text that enters the search engine is filtered. All conjunctions and prepositions are removed from it as irrelevant for the search. Then the system saves words or phrases from the text in the index. We should note that the words are not saved in their original form. They are converted in order to ease the search. Verbs, for example, are stored in the form of an infinitive, and only the root of a word can remain from nouns. It automatically takes into account all possible forms of verbs and noun cases, which significantly improves the search quality. The search engine also takes into account synonyms which are the words that are close in meaning. A text analyser in natural language can be used for this purpose.

Use of Different Options of Indexing Text

One of the possible implementations of the index may be the carrying out through the structure of an orderly tree. Searching in a sorted (for example, alphabetically) tree takes not a linear time, but a logarithm of the number of tree elements.

Along with this, it is necessary to decide how to transfer text from a website to a search engine and index it.

A search engine uses a program that checks Internet pages one by one in order to enter information about them into a search system database. It's called a search engine. It performs indexing of texts [6–7].

Indexing of texts can be external or internal.

1. *External Indexing*

There are search engines, also known as web spiders or crawlers. According to the principle of operation, the crawler resembles a normal browser; it goes to the page, reads its contents and follows all the links it finds on the page.

Search engine owners often limit the depth of crawler penetration into the site and the maximum size of scanned text. That is why excessively large sites may not be fully indexed by the search engine.

The operation of the crawleron the site can be configured using sitemap.xml and robots.txt files.

Sitemap.xml contains a sitemap, a list of links to all pages that must be indexed. Sometimes links to multimedia (pictures on the site) are added in sitemap.xml. Crawlers do indexing by replacing the text of these pictures (alt text).

Robots.txt may contain a list of addresses excluded from the index, restrictions on the frequency and timing of requests to the site.

The web page itself may also contain elements that the crawler should not index. Such parts are placed in the meta-tag <noindex> and the crawler ignores the contents of this tag [8].

2. *Internal Indexing*

Internal indexing assumes that site content is actively sent to the search engine, rather than waiting for the crawler to index the page.

Such indexing gives even more control over the indexed content, but also requires specific implementation.

If the search engine provides an API (Application Programming Interface), the indexing of the page is reduced to a simple indexing request. The text for indexing is sent in the body of this request. The address of the page where the text belongs to is also sent there.

Sending Text for Indexing

The time the text is sent for indexing can also vary.

CMS stores site content in a database. If we have a static site generator, the text for indexing can be sent by automated request at the time of publication of the site. In this case, the programmer should make changes to the site code, start the process of generating the final static files (html, js, css), copy the static files to the hosting server to send the contents of html files to the search engine using the API. Sending for indexing can take place on a timer (for example, once an hour).

For more complex sites, the process of sending text for indexing can become more complicated consequently. A special command (script) gives a possibility to connect to the CMS database, select the tables in which the content of the pages is stored, and send this content to the search engine.

Selection of Texts for Indexing Directly from the CMS Database

Connecting to the database and retrieving the required records is not a simple task by itself. Its implementation will already require the ability to program and deploy small autonomous programs.

Many CMS, such as WordPress, Magnolia CMS, store more than one version of the page. It is common practice to display the published version of the page and a draft of the latest changes. As soon as the webmaster considers the new changes ready to publish, the draft will be published and this version will be available to everyone. Then it should be included in the search index.

In the case of such architecture, a publication request is the most logical action when the content of the page should be sent to a search engine. By the way, when a de-publication (removal of a page from public access) is requested, a request should be sent to remove its content from the search index.

Choosing a Search Engine

The same web page can be indexed by different search engines. Thus, a site created with the help of a CMS can be externally indexed by Google's crawler, and internally – by its own search engine. Google will then use its own index and display links to pages of this site that are found by the crawler. The site itself can provide a form to search in the site, and use its own index of its own search engine.

A Google search form can also be embedded in the site. In this case, its own search engine on the site is not required. The display style of Google can be changed; so that the user of the site will not even guess that the search within the site is implemented using the Google index.

Google will use its own index and display search results only within the current site. This approach is fast and relatively easy to implement. It is only necessary to take into account the cost of connecting Google Search Engine (GSE) and the estimated number of search queries on the site. As the number of requests increases, so will the cost of using GSE.

The above stated solution cannot be used if the site contains confidential information, which for legal reasons should not be stored on third-party servers. In this case, all pages of the site should be excluded from the crawler index (using robots.txt), and the implementation of the search should be completely passed onto its own search engine, hosted on legally protected servers.

Consider the most common and popular complex search system Elasticsearch by Elastic [9].

Among the major sites that use Elasticsearch are Wikimedia, StumbleUpon, Quora, Foursquare, SoundCloud, GitHub and Netflix.

Amazon, IBM, Qbox and Elastic offer Elasticsearch to their subscribers as a manageable service. The system is hosted on the servers of these companies, but it is available to customers through the program interface (API).

In addition to full-text search, Elasticsearch is often used to search for and analyse event logs. For the convenience of compiling logs and sending them to the search engine, Elastic offers the Logstash product, and the Kibana platform gives a possibility to visualize and analyse data in the search index.

Elasticsearch, Logstash and Kibana are designed to be used as an integrated solution called Elastic Stack.

The Elasticsearch search engine is free, absolutely every user can installation his own servers [3, 4]. There is a docker image of Elasticsearch of different versions. For local development, this system can simply be run in docker. But on the server it can be deployed either by native installation on the server (usually a Linux machine), or the same docker image as for local development can be deployed in the Kubernetes cluster.

Implementing a Full-text Search with Elasticsearch

The given above analysis gives a possibility to choose the tools for the most effective implementation of full-text search. But many search engines that are implemented in Google, are inappropriate in the case of working with confidential information. The Elasticsearch system was chosen because it can be deployed on our own servers. In addition, it is extremely convenient and versatile and will ensure stable and reliable operation of the search algorithm. Google crawler data indexing is inappropriate in this solution because the crawler does not work together with Elasticsearch. Therefore, indexing is done internally.

So, let's describe the algorithm for implementing a full-text search in a full-fledged CMS using Elasticsearch.

1. One more step is added to the process of publishing or de-publishing a page in the CMS. This is the asynchronous sending of page content and its address to a specific service that is responsible for communicating with Elasticsearch. The process is marked as asynchronous in order not to block the normal operation of the CMS and not to be dependant on the availability of the indexing service. If the service does not respond and the page cannot be indexed, the page should be published anyway, and indexing may take place later.

2. Such an asynchronous interaction between the CMS and the indexing service can be provided by any Message Queue (solutions for processing the message queue) [10]. RabbitMQ, Apache Kafka, Amazon SQS (Simple Queue Service) belong to the most common and used by brokers messages.

3. The indexing service subscribes to messages from the Message Queue. The message contains the content of the page, its address and the command either include or exclude the page from the index.

4. The service generates a request to Elasticsearch to add or remove content from the index. If Elasticsearch is not available, the message in the queue is not marked as successfully processed, and after some time, the indexing request is retried.

If the service itself does not respond, the message is not read from the queue and it will remain there until the service is available again and will not process the message.

5. Once the page is added to the Elasticsearch index, the indexing or de-indexing process is complete.

6. A CMS front-end component is being developed, which is responsible for full-text search. This

component usually represents a form with a text box and a submit button. The component accepts the search query entered by the user, generates a query to Elasticsearch and reads the result.

7. Elasticsearch returns a list of pages that meet the search criteria, sorted by relevance. The CMS component, which reads the Elasticsearch response, displays a list of pages and links to them in a user-friendly form.

The advantages of this approach are obvious. The approach chosen is flexible and adapted to different website architectures. Asynchronous processing of indexing requests ensures the speed of performance and fault tolerance of the system. The system is divided into components flexibly enough to scale each individual component and the system as a whole.

There are also some disadvantages here, of course:

- the software implementation of the search service is necessary;
- additional costs for hosting Message Queue, indexing service and Elasticsearch itself;
- unavailability of the search function on the site in case of interruptions in the work of Elasticsearch, Message Queue or indexing service.

Conclusions

The article considers different approaches to the implementation of full-text search on the website, their advantages and disadvantages. Based on the results of the analysis, the most flexible and universal approach to the implementation of a full-text search system has been developed. The algorithm for implementing a full-text search using Elasticsearch is described. The advantage of the new approach is the asynchronous sending of page content and its address to a specific service that is responsible for communicating with Elasticsearch. It gives a possibility to unblock the normal operation of the CMS and beindependent on the availability of the indexing service. Queries to Elasticsearch are queued, and after a while there is another attempt made to process the indexing request. After searching is complete, Elasticsearch returns a list of pages that match the search criteria, sorted by relevance.

REFERENCES

1. “Polnotekstovyy poisk po saytu – bich sovremennogo interneta”, Habr. [online] Available at: <https://habr.com/ru/post/60551/> (Last accessed: 26.05.2021). (In Russian).
2. “Poisk podstroki v stroke”, Universitet ITMO. [online] Available at: <https://neerc.ifmo.ru/wiki/index.php?title=Poisk_podstroki_v_stroke> (Last accessed: 27.05.2021). (In Russian).
3. “Osnovy Elasticsearch”, Habr. [online] Available at: <https://habr.com/ru/post/280488/> (Last accessed: 18.11.2020). (In Russian).
4. “Stroim prodvinutyy poisk s ElasticSearch”, DOI. [online] Available at: <https://dou.ua/lenta/columns/building-advanced-search-with-elasticsearch/> (Last accessed: 18.12.2020). (In Russian).
5. “Obrabotka yestestvennogo yazyka”, Wikipedia. [online] Available at: <https://ru.wikipedia.org/wiki/Obrabotka_yestestvennogo_yezyka> (Last accessed: 8.12.2020). (In Russian).
6. “Poiskovyy robot”, Wikipedia. [online] Available at: <https://ru.wikipedia.org/wiki/ Poiskovyy_robot> (Last accessed: 19.10.2020). (In Russian).
7. “Standart ysklyuchenyy dlya robotov”, Wikipedia. [online] Available at: <https://ru.wikipedia.org/wiki/Standart isklyuchenij dlya robotov> (Last accessed: 23.04.2021). (In Russian).
8. “Noindex”, Wikipedia. [online] Available at: <https://ru.wikipedia.org/wiki/Noindex> (Last accessed: 02.10.2020). (In Russian).
9. “Elasticsearch”, Wikipedia. [online] Available at: <https://ru.wikipedia.org/wiki/Elasticsearch> (Last accessed: 11.09.2020). (In Russian).
10. “Ochered soobshcheniy”, Wikipedia. [online] Available at: <https://ru.wikipedia.org/wiki/ Ochered’_soobshcheniy> (Last accessed: 30.01.2021). (In Russian).

Received 16.11.2021

ЛІТЕРАТУРА

1. Полнотекстовый поиск по сайту – бич современного интернета. Habr. URL: https://habr.com/ru/post/60551/ (Last accessed: 26.05.2021).
2. Поиск подстроки в строке. Universitet ITMO. URL: https://neerc.ifmo.ru/wiki/index.php?title=Поиск_подстроки_в_строке (Last accessed: 27.05.2021).
3. Elasticsearch. Wikipedia. URL: https://ru.wikipedia.org/wiki/Elasticsearch (Last accessed: 11.09.2020).
4. Основы Elasticsearch. Habr. URL: https://habr.com/ru/post/280488/ (Last accessed: 18.11.2020).
5. Обработка естественного языка. Wikipedia. URL: https://ru.wikipedia.org/wiki/Обработка_естественного_языка (Last accessed: 8.12.2020).
6. Поисковый робот. Wikipedia. URL: https://ru.wikipedia.org/wiki/Поисковый_робот (Last accessed: 19.10.2020).
7. Стандарт исключений для роботов. Wikipedia. URL: https://ru.wikipedia.org/wiki/Стандарт_исключений_для_роботов (Last accessed: 23.04.2021).
8. Noindex. Wikipedia. URL: https://ru.wikipedia.org/wiki/Noindex (Last accessed: 02.10.2020).
9. Строим продвинутый поиск с ElasticSearch. DOI. URL: https://dou.ua/lenta/columns/building-advanced-search-with-elasticsearch/ (Last accessed: 18.12.2020).
10. Очередь сообщений. Wikipedia. URL: https://ru.wikipedia.org/wiki/Очередь_сообщений (Last accessed: 30.01.2021).

Надійшла 16.11.2021

Г.В. Ходякова, кандидат педагогічних наук, доцент, Миколаївський національний університет імені В.О. Сухомлинського, 54001, м. Миколаїв, вул. Шнеерсона, 11, Україна, khodiakovagalina@gmail.com

Н.В. Ходякова, провідний розробник, Ray Sono AG, Брудерхофштрасе 3, Мюнхен, 81371, Німеччина, nathalie.mk.ua@gmail.com

В.О. Поздєєв, доктор фізико-математичних наук, професор, Миколаївський національний університет імені В.О. Сухомлинського, 54030, м. Миколаїв, вул. Нікольська, 24, Україна, valer.al.pozdeev@gmail.com

ОРГАНІЗАЦІЯ ПОВНОТЕКСТОВОГО ПОШУКУ НА ВЕБ-САЙТІ

Вступ. При реалізації пошуку текстових фрагментів на сайті використовуються підходи різні за складністю та продуктивністю. Виникає також послідовність супутніх завдань: вибір варіанта індексації тексту, відправки тексту на індексацію, вибір текстів для індексації безпосередньо з бази даних *CMS*, вибір пошукової системи та інше. Ці підходи завжди забезпечують задовільні результати пошуку.

Мета. Опис існуючих рішень для повнотекстового пошуку на веб-сайті, їхніх переваг та недоліків. Розробка алгоритму повнотекстового пошуку за допомогою системи *Elasticsearch*.

Методи. Аналіз різних за складністю та продуктивністю підходів до реалізації повнотекстового пошуку на веб-сайті. Виявлення недоліків та вразливостей у більш примітивних підходах та розробка більш досконалих та складних алгоритмів, що усувають виявлені недоліки. Покрокова реалізація повнотекстового пошуку з використанням сторонніх систем.

Результати. Описано метод реалізації повнотекстового пошуку з використанням *Elasticsearch*. Перевагою нового підходу є асинхронне надсилання вмісту сторінки та її адреси у певний сервіс, який відповідає за комунікацію з *Elasticsearch*. Це дозволяє не блокувати нормальну роботу з *CMS* та не залежати від доступності сервісу індексації. Описаний у статті підхід є гнучким та адаптованим під різні архітектури веб-сайтів. Асинхронна обробка запитів на індексацію забезпечує високу швидкість виконання запиту та відмовостійкість системи.

Висновки. У статті розглянуто різні підходи до реалізації повнотекстового пошуку на веб-сайті, їхні переваги та недоліки. На підставі проведеного аналізу розроблено більш гнучкий та універсальний підхід до реалізації системи повнотекстового пошуку. Запропоновано рішення з покроковою реалізацією та налаштуванням просунутого повнотекстового пошуку з використанням *Elasticsearch*.

Ключові слова: *повнотекстовий пошук, алгоритм, обробка природної мови, пошуковий робот, сайт, індексація тексту, пошукова система.*