Bloupe

Yang, Jia Hao | León Quintana, Gerardo | Nagy, Jakub
Guerra Déniz, Irene | García Nuez, Raúl | Rivero Sánchez, Raúl

University of Las Palmas de Gran Canaria, Las Palmas de Gran Canaria, Spain.

Abstract

The digital age has witnessed an unprecedented proliferation of information, and open-access repositories like the Gutenberg project have made an extensive collection of books available to the public. To facilitate the retrieval of relevant content from this vast literary treasure trove, this paper presents the design and implementation of a search engine in Java specifically tailored for indexing and searching books obtained from the open-source Gutenberg project.

Our Java-based search engine is designed to efficiently index and retrieve information from a large corpus of books, including classical literature, historical texts, and more. In this work, we provide a comprehensive overview of the architecture and components of our system. The system encompasses web crawling, text analysis, and indexing techniques optimized for books, such as metadata extraction and book-specific content analysis.

This paper also includes an evaluation of the search engine's performance, including indexing speed, retrieval times, and the quality of search results when applied to the Gutenberg project's books. Comparative analyses with other search engines further illustrate the effectiveness and efficiency of our system in the context of literary content retrieval.

In conclusion, our Java-based search engine, tailored for open-source Gutenberg project books, provides a practical and efficient means of searching through this vast collection of literary works. This contribution not only aids researchers and readers in accessing classical and historical texts but also offers insights into the development of specialized search engines for literary and academic purposes.

Introduction

In the current digital age, the volume of information available online has reached exponential proportions. This wealth of data has necessitated sophisticated processing and analysis techniques to extract meaningful insights. In this context, Project Gutenberg, an initiative aiming to digitize and archive an extensive collection of public domain books, stands as an invaluable source of literary information. However, to fully utilize this treasure trove of knowledge, efficient and precise search tools within this vast digital library are essential.

This paper presents an innovative approach to address this challenge using Big Data technologies and the Java programming language. We propose an inverted indexing system that leverages crawling capabilities to gather content from Project Gutenberg and data cleaning techniques to prepare texts for analysis. The crux of our approach lies in constructing an inverted index, a fundamental component of modern search engines, enabling users to find specific books based on keywords.

This work details the design and implementation of this system, highlighting challenges encountered during development and proposed solutions. Additionally, results obtained through rigorous testing are presented, demonstrating the efficacy and accuracy of our approach in searching for information within the extensive repository of Project Gutenberg. This study not only represents a significant advancement in the field of indexing large volumes of literary data but also showcases the feasibility and utility of applying Big Data technologies in the preservation and accessibility of historical and cultural knowledge.

Methodology

The project has been implemented using fourth modules, to divide the responsibilities of manufacturing the search engine. In this section we will explain the structure of each module, how they work and their main characteristics. This search engine is composed of fourth modules: Crawler, Indexer, Query and a shared module.

Crawler

Book Retrieval Process

The fundamental aim of this module revolves around retrieving books from the Gutenberg Project. Leveraging the `GutenbergProjectBookDownloader` class, this process is orchestrated to establish a comprehensive datalake housing the downloaded literary resources.

Integration with Google Cloud

Subsequently, this datalake undergoes migration to the Google Cloud platform via the GoogleCloud-DataLakeBookUploader. This strategic shift optimizes scalability and facilitates the creation of multiple crawlers, thereby enhancing operational efficiency.

ArtemisMQProducer for Coordination

To streamline coordination and bolster indexing efficacy, the `ArtemisMQProducer` class is employed. Its utilization of the ArtemisMQ broker enables seamless management of download notifications through an efficient message queue system.

Indexer

Consumption of Queue Messages

The controller initiates an infinite loop to constantly listen for messages in the queue, utilizing the `ArtemisMQBooksConsumer` class to consume messages from the Apache ActiveMQ Artemis queue. Subsequently, it retrieves the file path from the consumed message and uses the `GoogleCloudDataLakeManager` class to download the file content from Google Cloud Storage.

Fetching File Content

Once the file is downloaded, the `FileContentManager` class is used to obtain its content. Similarly, the same class is employed to fetch the metadata portion of the file content. Metadata information is processed using the `MetadataManager` class, which creates a `Book` object.

Storing the Book Object in the Database

Persistence of the `Book` object is carried out using the `PostgreSQLBookRepository` class to store it in the PostgreSQL database.

Indexing File Content

File content indexing is performed using the `Indexer` class. A list of `IndexedWordResult` objects representing indexed words and their frequency in the book is obtained.

Adding Words to the Datamart

To add the results of indexed words to the Datamart, the `HazelcastDataMartManager` class is used. Each entry in the Data Mart contains information about the frequency of the word in a specific book.

Waiting for Another Message in the Queue

After completing the processing of a message, the controller returns to the beginning of the loop to wait and consume another message from the queue.

Query

Requirements Review:

- **Main Functionality:** Conduct search queries on the API. Retrieve information about books and their frequency of appearance for a keyword.
- **Scalability:** Ensure that the system is scalable to handle multiple simultaneous queries from different computers.

Implementation:

- `PostgreSQLBookRepository`: Utilizes HikariCP to manage the connection pool to the PostgreSQL database. Implements logic to search for books by ID.
- `GoogleCloudSearchRepository`: Uses Google Cloud credentials to access a storage service and retrieve information on the frequency of keywords.
- `HazelcastSearchRepository`: Implements a caching mechanism using Hazelcast, efficiently storing and retrieving data.
- `SearchServiceImpl`: Normalizes the keyword to lowercase for consistency. Retrieves search results from the `SearchRepository` implementation. Combines results with detailed book information using the `BookRepository` implementation.

Deployment:

The code is deployed as a web service using Spark, ensuring availability for conducting search queries from different computers.

Shared Module: `SharedDataMartEntry` Class

The `SharedDataMartEntry` class within the `sharedmodule` encapsulates a data structure representing an entry in the shared data mart. It holds information about a book's unique ID and the frequency of appearance of a keyword in that book.

- **Attributes:**
  - `bookId` (`int`): Unique ID of the book.
  - `appearance` (`int`): Frequency of appearance of a keyword.
- **Methods:**
  - Constructor for initialization.
  - Methods for accessing and modifying `bookId` and `appearance`.

Experimentation

In this section, we will delineate the experimental procedures conducted to validate and ensure the appropriate functionality of our application, while assessing its performance acceptability. Initially, the primary objective of the project was scrutinized, namely, the determination of its compatibility within a cluster of computers. To fulfill this, a cluster of computers was employed for experimentation. It is noteworthy to mention that conducting experiments within the university's computer network was precluded due to its secure nature, and the architecture of our application necessitates connectivity with a remote server. Despite this constraint, efforts were made to devise alternative testing methodologies.

The application was executed on personal computers, configured to emulate a scenario with one host and three requestors. While the application exhibited impeccable functionality, a notable concern arose regarding memory consumption, attributed to the simultaneous execution of multiple processes on a single machine. The load balancer of NGINX demonstrated commendable efficacy, and the processes associated with downloading and indexing books operated seamlessly. It is pertinent to note that upon termination of all processes, the Datamart is purged from memory, necessitating a complete restart from an initial state. These observations contribute to the comprehensive understanding of the application's behavior in a controlled environment.

Conclusion

The development of a specialized Java-based search engine tailored for the Gutenberg project represents a significant stride in literary content retrieval within the digital era. By leveraging Big Data technologies and sophisticated methodologies, this search engine offers a practical and efficient means of navigating through the extensive collection of classical and historical texts available through Project Gutenberg.

The paper outlines a meticulous methodology encompassing the design and implementation of the search engine, delineating the intricate structure and functionalities of its modules: Crawler, Indexer, Query, and the Shared Module. Through integration with Google Cloud, utilization of message queue systems for coordination, and leveraging databases and indexing techniques, the system showcases a comprehensive approach to efficiently handle the retrieval, processing, and indexing of literary resources.

Experimental validation conducted under controlled conditions provides insights into the system's functionality and performance. Despite constraints regarding experimental environments, the system demonstrated robust functionality, albeit with concerns about memory consumption in scenarios involving multiple simultaneous processes on a single machine. The efficacy of load balancers and the seamless operation of critical processes highlight the system's reliability and efficiency in practical settings.

In conclusion, this Java-based search engine not only addresses the challenge of sifting through extensive literary repositories but also underscores the viability of employing advanced technological frameworks in preserving and facilitating access to historical and cultural knowledge. The system's adaptability, efficiency, and comprehensive functionality mark a substantial contribution to the domain of specialized search engines tailored for academic and literary purposes.

Future work

Future work in this domain holds immense potential for advancing the capabilities and scope of the Java-based search engine tailored for the Gutenberg project. Building upon the foundation laid by this research, several avenues for further exploration and enhancement emerge:

- **Optimized Resource Utilization:**Addressing concerns about memory consumption during simultaneous processes to optimize resource utilization, possibly through resource allocation strategies.
- **Scalability Testing:** Conducting extensive scalability tests to assess the system's performance under varying loads and with larger datasets.
- **Semantic Search:** Introducing natural language processing techniques to enable semantic search capabilities, allowing for more context-aware and nuanced queries.
- **Integration with External Databases:** Exploring integration with other literary databases or repositories beyond Project Gutenberg, broadening the scope of accessible content.

The future trajectory of this research involves a holistic approach encompassing technical advancements, user-centric enhancements, and collaborative efforts, thereby solidifying the search engine's position as an indispensable tool for accessing and exploring vast repositories of literary and historical texts.