Solr Connector

Team FML

Problem Statement 4:
Apache Solr Data Indexing & Metadata Extraction

GitHub

Introduction

- Apache Solr is a powerful, open-source search engine built on top of Apache Lucene, designed to efficiently index and search structured data at scale. It is widely used in various applications for its robust features, including faceted search, filtering, and auto-suggest capabilities, making it an ideal choice for enhancing user experience in web applications.

- The challenge presented involves leveraging Solr's capabilities to index JSON documents and extract metadata using Python. This task requires setting up an Apache Solr instance, creating a core to manage indexed data, defining a schema for structured JSON documents, and developing a Python connector to interact with Solr. The Python script should accept user inputs for connection details and extract core-level metadata, including core name, index name, index size, and the number of documents. Additionally, it must retrieve two randomly selected documents and extract their unique IDs along with first-level fields and values, ignoring deeply nested structures.

- This project aims to demonstrate how Solr can be utilized for efficient data indexing and metadata extraction, showcasing its potential in handling both nested and non-nested JSON documents. By successfully implementing this solution, we can highlight Solr's scalability and flexibility in managing diverse data formats, making it a valuable tool for applications requiring advanced search functionalities.

4. Problem Statement

Apache Solr Data Indexing & Metadata Extraction

The challenge: to install, configure, and utilize Apache Solr for data indexing and metadata extraction using Python.

Key Requirements:

- Install and configure Apache Solr.
- Create a core and define a suitable schema.
- Develop a Python connector.
- Extract metadata and sample documents.
- Handle nested JSON by considering only the first level of keys.

Steps done to achieve the solution

- We began by **installing Apache Solr** on our system, which involved downloading and setting up the Solr server. Once installed, we created a core to store and manage the indexed data. This core was specifically designed to handle both nested and non-nested JSON documents.
- Next, we defined a schema for indexing structured JSON documents. This schema was crucial as it determined how Solr would interpret and store the data from the JSON documents. We ensured that the schema was flexible enough to accommodate both nested and non-nested structures, focusing on extracting metadata from the first level of keys in each document.
- After setting up the core and schema, we added both nested and non-nested JSON documents into the core. This step involved using Solr's API to upload the documents, ensuring that they were properly indexed according to the defined schema.
- We then developed a Python connector that accepted user inputs for Solr connection details, including hostname, port number, core name, username, and password. This script utilized the Solr Python client to connect to Solr and perform operations such as fetching metadata and retrieving sample documents.
- To extract index metadata, we used the Python script to fetch and display core-level metadata, including the core name, index name, size of the index in bytes, and the number of documents in the index. Additionally, we retrieved two randomly selected documents from the index and extracted their unique document IDs along with the first-level fields and values, ignoring any deeply nested structures.
- Finally, we exported the extracted metadata and sample documents to a JSON file named **solr_metadata.json**, ensuring proper formatting and indentation for readability. This file served as the final output of our solution, providing a structured overview of the extracted metadata and sample documents.

5. Final Result

```
PS D:\Programming\PICT-HACKATHON> python solr_connector.py
Enter Solr hostname (default: localhost): localhost
Enter Solr port number (default: 8983): 8983
Enter Solr core name (default: my_core): my_core
{'core name': 'my_core', 'index name': 'my_core', 'size of index': 7220, '# documents': 5}
Solr metadata saved
PS D:\Programming\PICT-HACKATHON>
```

Working of solr connector

Drive link for video of working: [Video]

Sample output of JSON file

5. Conclusion

- Our solution successfully achieved the objective of setting up Apache Solr, indexing JSON documents, and extracting metadata using Python. We effectively installed Solr, created a core, defined a schema for JSON documents, and developed a Python script to connect to Solr and extract core-level metadata. The script also retrieved two randomly selected documents, extracting their unique IDs and first-level fields and values, while ignoring deeply nested structures.

- The solution worked as intended, demonstrating Solr's capability to handle both nested and non-nested JSON documents efficiently. By using Python to interact with Solr, we were able to automate the process of metadata extraction, making it scalable for large datasets. The output was exported to a JSON file, ensuring readability and ease of use for further analysis.

- Overall, this project showcases the power of combining Apache Solr with Python for data indexing and metadata extraction tasks, highlighting its potential for applications requiring robust search and data management capabilities.

THANK YOU

Arya Lingayat

Anurag Muley

Rohan Mali

Mustansir Rangwala