Journal Homepage: - www.journalijar.com

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH (IJAR)

Article DOI: 10.21474/IJAR01/16920
DOI URL: http://dx.doi.org/10.21474/IJAR01/16920

RESEARCH ARTICLE

ENHANCING SEARCH CAPABILITIES: EXPLORING LUCENE AND SOLR TECHNIQUES FOR IMPROVED SEARCH PERFORMANCE

Prathyush Krishnen J.[1] and S. Yamuna[2]

1. Research Scholar, Department of Computer Science and Engineering, Meenakshi Sundararajan Engineering College, Chennai.
2. Assistant Professor, Department of Computer Science and Engineering, Meenakshi Sundararajan Engineering College, Chennai.

Manuscript Info

Manuscript History
Received: 20 March 2023
Final Accepted: 22 April 2023
Published: May 2023

Abstract

In today's digital age, data is everything. Dealing with and accessing a lot of data is a common task for almost every individual, be it for personal or for professional purposes and they expect applications to provide fast and accurate search capabilities to access and view their data. But all this data becomes meaningless if it cannot be accessed easily when needed or within a certain amount of time. This study aims to implement a powerful and customizable search feature in applications using the open-source search technologies, Lucene and Solr. The system will use Lucene's efficient indexing and querying capabilities to create an inverted index of the application's content, enabling fast and accurate full-text search. Solr will provide a web interface for the search functionality, allowing users to easily perform searches and filter results by various criteria such as date, author, category, and more. By incorporating Lucene and Solr, the system will be able to overcome the previous mentioned issues by providing search results significantly faster and deliver a robust, fast, and user-friendly search experience for applications that can be easily integrated into any application.

Introduction:-

Search technology is an important field that focuses on developing algorithms and techniques for retrieving information from large datasets. It encompasses a wide range of subfields, including information retrieval, natural language processing, machine learning, and data mining. Information retrieval, in particular, is a key subfield that focuses on developing techniques for retrieving relevant information from large collections of data, such as text documents, images, and audio files. The concept of search technology and information retrieval is particularly important in today's modern digital age, where large volumes of data are generated and stored every day. Search technology helps users to find the information they need quickly and efficiently, improving productivity and efficiency.

However, this field also presents a number of challenges, particularly as datasets continue to grow in size and complexity. By implementing a search feature using Lucene and Solr, we will be working within this important domain, applying advanced algorithms. Lucene works based on the concept of Inverted Indexing which can

Corresponding Author:- Prathyush Krishnen J
Address:- Department of CSE, Meenakshi Sundararajan Engineering College, Chennai, India.

drastically reduce the amount of time taken for applications to search for data and also reduce the load on the system as the system just needs to search through the generated indices instead of the entire data. Solr then helps in searching these generated indices quickly. Apart from just directly searching the indices, Solr also provides other features such as facets and highlighting which makes it an idea tool for building an advanced search engine.

Problem Statement:-

Developing an efficient and effective search feature for applications can be a challenging task, especially when dealing with large volumes of unstructured data. Traditional keyword matching and exact string matching methods are often insufficient to provide relevant search results for users. Therefore, the problem this project aims to address is to create a Lucene and Solr-based search engine for applications that can provide fast and accurate search results for users. The search feature should be able to handle large volumes of structured and unstructured data, and be flexible enough to support a wide range of search queries and filters.

Literature Survey:-

A. Tao Qiu, Xiaochun Yang*,Member,IEEE, Bin Wang, and Wei Wang (2020), "Efficient Regular Expression Matching Based on Positional Inverted Index", IEEE Transactions on Knowledge and Data Engineering (Volume: 34, Issue: 3, 01 March 2022))

The paper proposes a new method for whole-database retrieval of general relational databases using the Lucene search engine. The proposed method is able to provide efficient and flexible search capabilities for relational databases, allowing users to perform keyword searches and complex queries over the entire database.

B. Mehdi Kargar (2022), "Search System over e-Commerce Data for Business Users", 2022 International Conference on Engineering & MIS (ICEMIS)

This paper proposes a search system over e-commerce data that is specifically designed for business users. The system uses Apache Solr as the search engine and includes features such as faceted search and dynamic filters.

C. Xiaofei Yao, Jin Li, Yaodong Tao, Shenglong Ji (2022), "Relational Database Query Optimization Strategy Based on Industrial Internet Situation Awareness System", 2022 7th International Conference on Computer and Communication Systems (ICCCS)

This paper proposes a new query optimization strategy for industrial internet situation awareness systems. The proposed strategy takes into account the real-time processing needs, data heterogeneity, and dynamic data updates of these systems. It includes a data model, query execution engine, and query optimization module to select the best execution plan.

D. Dawei Jin, Gang Chen, Wenning Hao, Liu Bin (2020), "Whole Database Retrieval Method of General Relational Database Based on Lucene", 2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)

This paper proposes a whole-database retrieval method of a general relational database based on Lucene, which provides full-text search capability over large datasets in a relational database. The proposed method is designed to extract and index text data from all tables in a database, enabling users to perform full-text searches across the entire database.

E. Li Ran, Wu Xiaojin (2020), "Full-text Retrieval System for Structured Data of University Archives", 2020 3rd International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE)

This paper proposes a full-text retrieval system for structured data of university archives. The system uses Apache Lucene and Solr to provide efficient and accurate retrieval of information from structured data. The system includes features such as faceted search, filtering, and sorting, and can be used to search across multiple collections of data.

F. Wei Ji (2020), "Research and Application of Information Data Retrieval System in Station Based on Lucene Technology", 2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA).

The paper discusses the development of an information retrieval system for a train station using Lucene technology. The system was designed to provide real-time and accurate information to train passengers, including train schedules, arrival and departure times, and delay notifications.

G. Zheng Youzhuo, Fu Yu, Zhang Ruifeng, Hao Shuqing, Wen Yi (2020)," Research on Lucene Based Full-Text Query Search Service for Smart Distribution System", 2020 3rd International Conference on Artificial Intelligence and Big Data (ICAIBD).

This paper presents a Lucene-based full-text query search service for a smart distribution system. The system uses the Apache Lucene search engine to provide efficient and accurate retrieval of information from distributed databases. The proposed search service supports advanced features such as faceted search, filtering, and sorting, and can be easily integrated into existing smart distribution systems.

Existing System With Drawbacks:-

Before Lucene, search technologies and methodologies were generally based on simple keyword matching or exact string matching. Some of the earliest search systems were based on manual indexing of documents using controlled vocabularies or classification schemes. This approach required extensive manual effort and was limited in its ability to capture the nuances of natural language and the diversity of user queries. As computers became more powerful and more sophisticated search algorithms were developed, other search methodologies emerged. One early search methodology was the vector space model, which represented documents and queries as vectors in a multi-dimensional space, with the distance between vectors indicating the relevance of a document to a query. Another methodology was the Boolean model, which used Boolean logic to combine search terms and retrieve relevant documents.

In the late 1990s, probabilistic models such as Okapi BM25 were developed, which took into account statistical patterns in the occurrence of search terms to calculate relevance scores for documents. These models were particularly effective at dealing with large and diverse collections of documents, but they still relied primarily on keyword matching and exact string matching. Overall, the search technologies and methodologies before Lucene were less sophisticated and less effective than modern search systems, but they paved the way for the development of more advanced and more flexible search technologies like Lucene and Elasticsearch.

Architecture:-

The main idea behind the searching feature is to use Lucene to generate indexes for data and use Apache Solr to search through the generated indexes to find the relevant search results from the query. The user is not expected to input a complete sentence or phrase in order to find the required data/record but rather the search feature is efficient enough to displays all relevant results that are related to the user query even if they are not a perfect match.

Lucene uses an inverted index to map terms to documents, which is a data structure that maps terms to the documents that contain them. This hence allows for faster searching of large sets of data. The search results can be ranked based on relevance scores, which are calculated using various factors such as term frequency and inverse document frequency. Solr is a search platform that is built on Lucene which provides a set of APIs that allows developers to easily add search functionalities to their applications.Solr also has additional search features like faceting, highlighting and more which makes it an idea tool for building advanced search features.

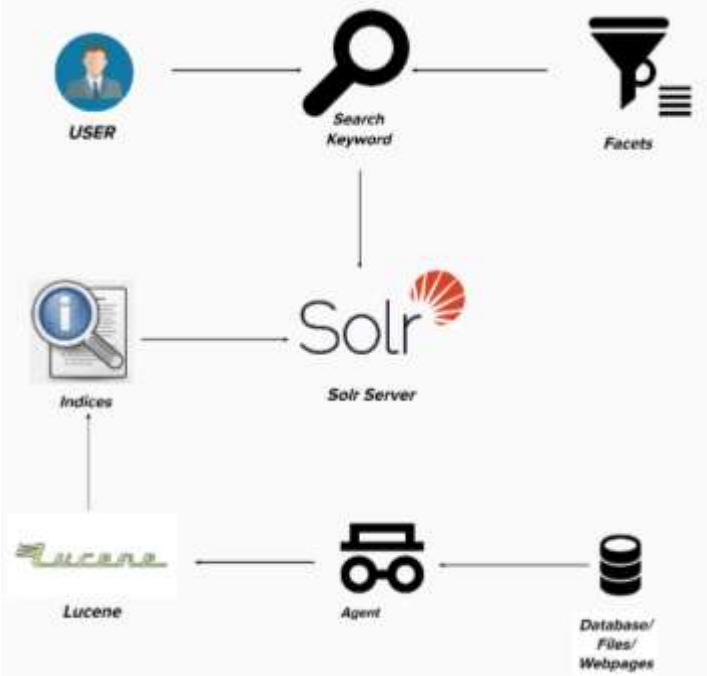

Figure 1:- Project Architecture.

The search implementation using Lucene and Solr starts by gathering data from a variety of sources like databases, files, webpages or by manual input and is sent to the indexing server for indexing. It is then indexed using Lucene by a series of steps like parsing and analyzing and the indexes are stored in a server for faster access.The user enters a search query that is then sent to the searching server to search the indexes.With the search query, the searching server, Solr, searches the indices for relevant search results.The results generated from the searching server are then displayed back to the user.

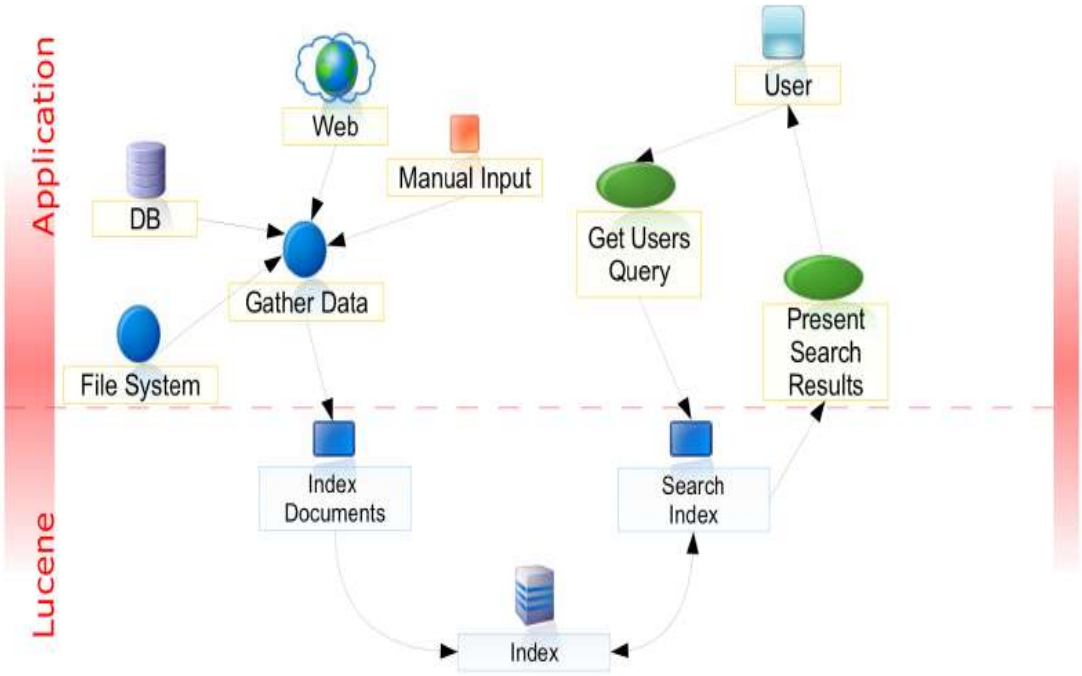

Figure 2:- System Architecture.

Proposed Solution:-

Inverted Index

In its basic form, an inverted index consists of postings lists, one associated with each term that appears in the collection. A postings list is comprised of individual postings, each of which consists of a document id and a payload information about occurrences of the term in the document. The major benefit of using an inverted index is to speed up the searching. This is done by not searching the whole document set, instead searching only the index file. Index file may also contain the ranking details which is often necessary to present the best result to the user. The major steps in this are:

1. Tokenize the text, turning each document into a list of tokens.
2. Do linguistic pre-processing, producing a list of normalized tokens, which are the indexing terms.
3. Index the documents that each term occurs in by creating an inverted index, consisting of a dictionary and postings.

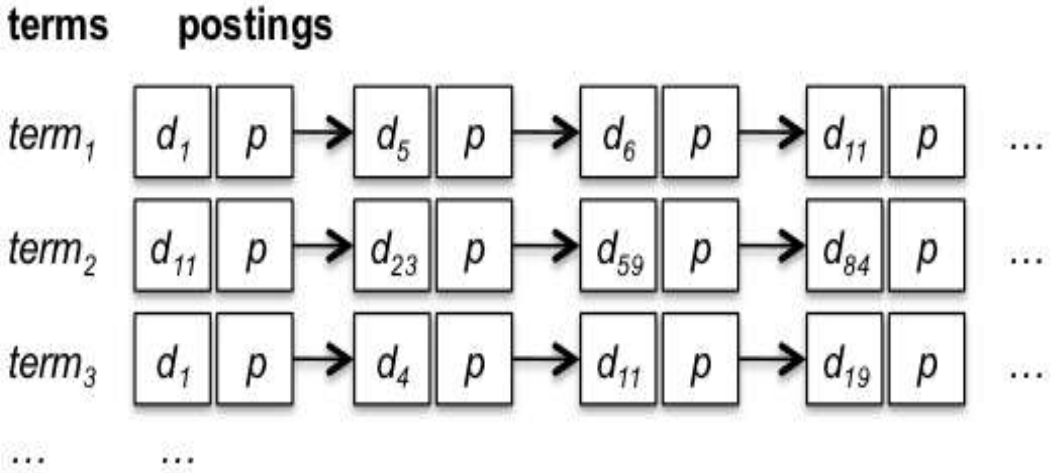

Figure 3:- Inverted Indexing.

Methodology:-

The Lucene module is responsible for indexing and searching text-based data using the Lucene library. This module provides functionality for creating and updating indexes based on data from various sources, including databases and files. The Lucene module uses a combination of tokenization, analysis, and indexing techniques to create an inverted index that allows for fast and efficient searching of large amounts of text-based data.

Parser

The parser component in Lucene is responsible for converting input data into a format that can be indexed by Lucene. This involves breaking the input data down into individual tokens, which are then analysed and added to the Lucene index. The parser in Lucene can handle a variety of input formats, including plain text, HTML, XML, and JSON. It also provides support for custom input formats through the use of plugins and extensions. The parser works by first breaking the input data down into individual characters, which are then grouped into tokens based on a set of predefined rules. For example, whitespace, punctuation, and other delimiters can be used to separate tokens.The parser also provides support for filtering out irrelevant or unwanted data, such as stop words or special characters. This can help improve the quality and relevance of search results by reducing the amount of noise in the index.

Analyser

The analyser component in Lucene is responsible for processing individual tokens and extracting useful information from them. This involves applying a series of analysis techniques to the tokens, such as stemming, stopword removal, and synonym mapping. The analyser in Lucene can be customized to suit the specific needs of your project. For example, a stemmer can be used to reduce words to their root form, or a stopword filter to remove common words like "the" or "and". Custom filters can also be created to handle more complex analysis tasks, such as entity recognition or part-of-speech tagging.

The analyser works by taking a stream of tokens from the parser and applying a series of filters to each token in turn. Each filter can modify the token in some way, such as removing characters, changing case, or replacing synonyms. The resulting tokens are then added to the index, where they can be searched and retrieved using Solr's search API.

Indexing

Once the text is analyzed, the terms and their positions are added to an index structure that allows for fast searching and retrieval of documents based on the search query. The index structure consists of several components, including a lexicon or dictionary of all the unique terms in the corpus, as well as inverted index data structures that map each term to the documents that contain it. In addition, the index includes various metadata fields like document ID, document boost, and term frequency, which are used to rank and score the search results.

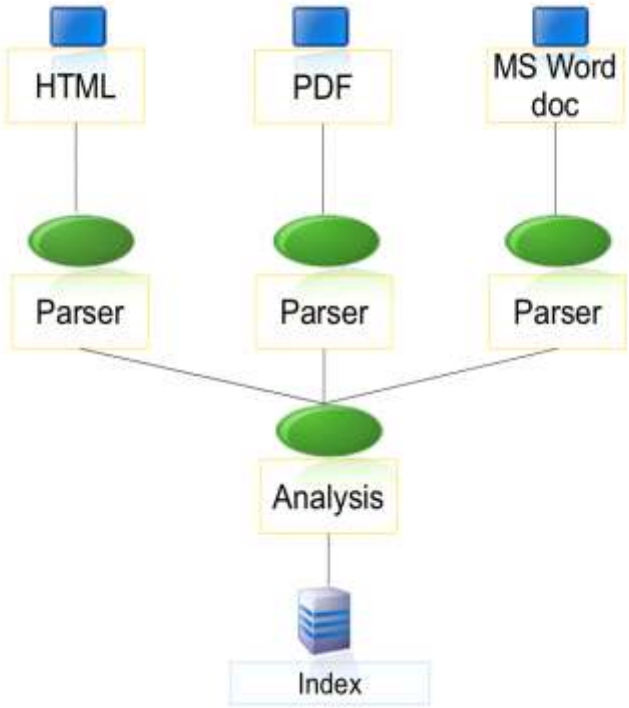

Figure 4:- Indexing Flow.

Searching

The Solr module is a search platform that builds on top of Lucene and provides additional features such as faceted search, hit highlighting, and real-time indexing. It also provides a RESTful API for integrating with other applications and systems. The Solr module uses the Lucene index as its underlying data source, and provides a layer of abstraction that allows for easy configuration and management of the search platform.

When a user initiates a search query in Solr, the query is first parsed to extract the relevant terms and operators. The query may then be transformed or modified based on various rules and settings specified in the Solr configuration, such as applying synonyms or stemming. Solr then executes the search against the Lucene index, utilizing query optimization techniques like caching and parallelization to improve search performance. Once the search is executed, Solr retrieves the search results from the Lucene index and sorts them based on their relevance to the query, as well as computing any relevant facets or aggregations. The search results are then formatted into a response that can be sent back to the user, which can be customized based on their preferences and include metadata like the total number of results and search execution time. Finally, Solr delivers the search results back to the user via the web interface or API, allowing them to interact with the results and refine their search query as necessary.

Significance:-

A search engine that is built using Lucene and Solr can provide

1. **Powerful Information Retrieval:** Lucene is a high-performance, full-text search library that provides indexing and searching capabilities. It allows applications to efficiently search and retrieve information from large volumes of textual data.
2. **Scalability and Performance:**Solr is an enterprise search platform built on top of Lucene. It adds features like distributed searching, faceted navigation, caching, and more. Solr enables horizontal scaling, ensuring that your search engine can handle increasing data volumes and user queries while maintaining high performance.
3. **Integration with Existing Systems:** Lucene and Solr provide APIs and connectors that facilitate integration with existing applications and data sources. They support various data formats and protocols, making it easier to ingest and index data from different sources like databases, files, or web services.
4. **Text Analysis and Language Support:** Lucene and Solr provide extensive support for text analysis, allowing you to perform tokenization, stemming, synonym expansion, and other linguistic processing tasks. They also support multiple languages, making it easier to build multilingual search applications.
5. **Rich Query Capabilities:** Lucene and Solr offer a wide range of query capabilities, including support for advanced search features like Boolean queries, phrase searches, fuzzy searches, wildcard searches, and relevance scoring. This allows you to build robust search functionalities within your applications.

Results:-

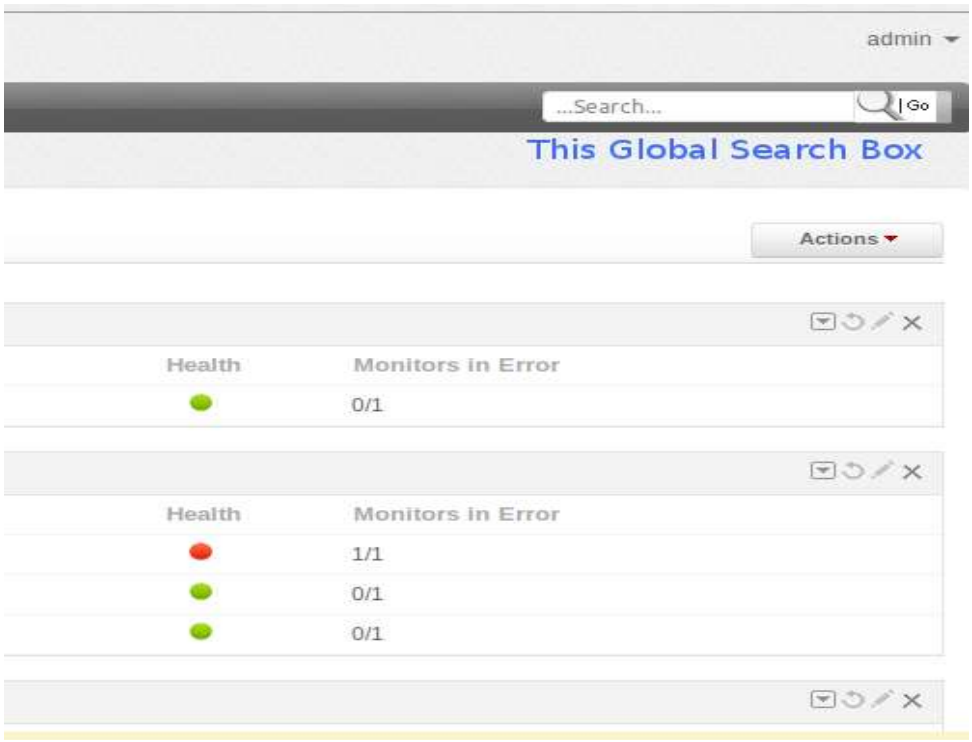

Figure 5:- Global Search Bar.

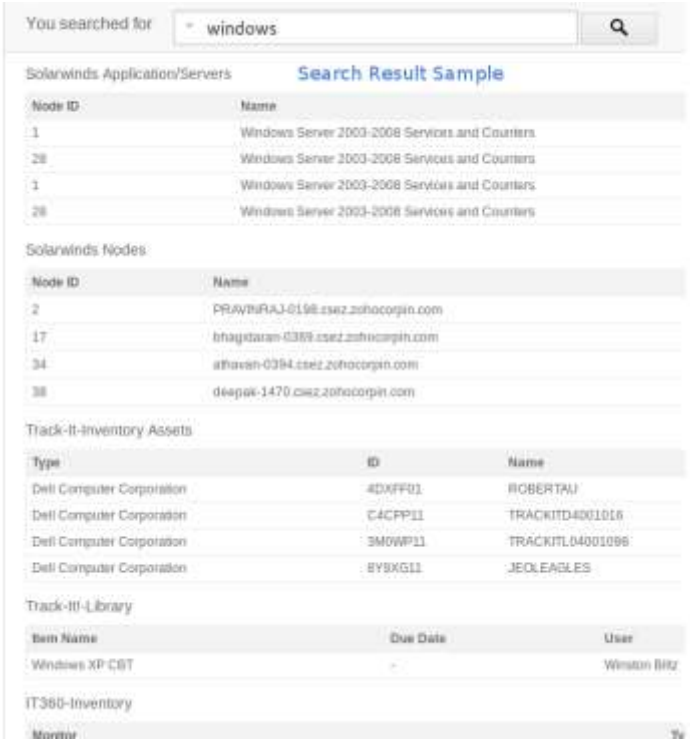

Figure 6:- Sample Search Result.

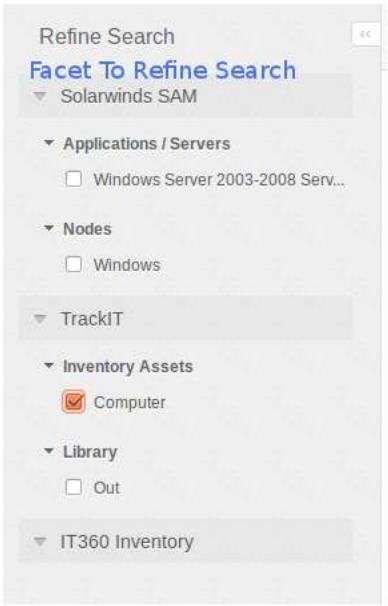

Figure 7:- Facet Selection.

Conclusion:-

Hence the Search Engine using Solr can be integrated with any product which uses structureddata as its mainstorage engine. Products like CRMs, data management applications and more generates huge data sets as it's output and isnot easy to find the required information whenever needed. And after all there is no point ofinformation, if it can be used at right time. The search engine using Lucene and Solr implements all the features required to provide search for structured data. Search engine implementation required implementing data crawling, data indexing, and providing searchfeature to the user.

Data Indexing is done to provide faster lookup and hence optimal performance for the search canbe achieved. Facet Configuration is very much useful to refine the search result.Following were the special features, which supersedes the search implementation:

Global Search Box:
It is the input field which allows the user to enter the keyword he/she would like to search and start searching.There exists only one such field to search across all the modules. This search box gives the unified search result from all the modules.

Filtering Based On Criteria:
Facet is used to provide filtering option to refine the search result.

Future Enhancements:-
Additionally, the project could be enhanced by adding features to improve the overall search experience for users, such as autocomplete suggestions, search-as-you-type functionality, or visual search interfaces. Scalability and performance optimizations may also be necessary for handling larger volumes of data and search traffic. A few of these are mentioned briefly below

Improved search relevancy:
One area for improvement in your project is the relevancy of search results. You could explore techniques like machine learning-based ranking models, query expansion, or relevance feedback to improve the accuracy and relevance of search results.

Advanced query parsing:
While Lucene provides a powerful query parsing engine, there may be cases where more advanced query parsing capabilities are needed. You could explore integrating more advanced query parsing libraries or techniques into your project to enable users to construct more complex and powerful search queries.

Enhanced search experience:
You could explore adding additional features to enhance the search experience for users, such as autocomplete suggestions, search-as-you-type functionality, or visual search interfaces.

Scalability and performance:
If your project is expected to handle a large volume of data or search traffic, you may need to explore techniques for scaling and optimizing performance, such as sharding the index or utilizing caching and load balancing.

References:-

[1] Dawei Jin, Gang Chen, Wenning Hao, Liu Bin (2020), "Whole Database Retrieval Method of General Relational Database Based on Lucene", 2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)
[2] Li Ran, Wu Xiaojin (2020), "Full-text Retrieval System for Structured Data of University Archives", 2020 3rd International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE)
[3] Mehdi Kargar (2019), "Exploring Structured Information Systems: A Keyword Search Approach", 2019 4th International Conference on Communication and Information Systems (ICCIS)
[4] Mehdi Kargar (2022), "Search System over e-Commerce Data for Business Users", 2022 International Conference on Engineering & MIS (ICEMIS)
[5] Sanu Lakhara, Dr. Nidhi Mishra (2017), "Desktop Full-Text Based Search based on Lucene: a Review", 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)
[6] Tao Qiu, Xiaochun Yang*,Member,IEEE, Bin Wang, and Wei Wang (2020), "Efficient Regular Expression Matching Based on Positional Inverted Index", IEEE Transactions on Knowledge and Data Engineering (Volume: 34, Issue: 3, 01 March 2022)
[7] Wei Ji (2020), "Research and Application of Information Data Retrieval System in Station Based on Lucene Technology", 2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA).
[8] Xiaofei Yao, Jin Li, Yaodong Tao, Shenglong Ji (2022), "Relational Database Query Optimization Strategy Based on Industrial Internet Situation Awareness System", 2022 7th International Conference on Computer and Communication Systems (ICCCS)

[9] Xiaomei Chen, Lizhen Xu (2016), "An Educational Resource Retrieval Mechanism Based on Lucene and Topic Index", 2016 13th Web Information Systems and Applications Conference (WISA)
[10] ZhengYouzhuo, Fu Yu, Zhang Ruifeng, Hao Shuqing, Wen Yi (2020)," Research on Lucene Based Full-Text Query Search Service for Smart Distribution System", 2020 3rd International Conference on Artificial Intelligence and Big Data (ICAIBD).