An Intelligent Spatial Aware Search Engine Using Lucene and Solr

Thirunavukkarasu K[1], Priyank Tripathi[2], Deepak Upadhay[3] and Dr.Manoj Wadhawa[4]
[1,2,3]Galgotias University, India, [4]Echelon Institute of Technology, India

***Abstract-* All forms of human activity are rooted in geographic space in some or the other way. Apparently, many web resources include references to geographic context. Spatial Web Portals have drastically improved the sharing and exchanging of Earth Science data, information and services. We have collected large amount of geospatial data, metadata and web data. It has been cataloged and kept available through SWPs to serve a broad user community. Most search engines in SWPs are used in this paper based on keyword matching, which cannot effectively 'understand' the meaning of users' queries, especially when a user has limited Earth science knowledge. So finding the needed accurate information from a variety of geospatial resources becomes a big challenge. We have proposed a semantic search engine using lucene and solr in this paper to make the spatial-aware search more intelligent and accurate.**

Keywords: SWPs, Lucene, Solr, GPSE

I. Introduction (*Heading 1*)

This World Wide Web is expanding every second just like the universe. The size of the World Wide Web (The Internet) The Indexed Web contains at least 4.73 billion pages (Friday, 13 November, 2015).[15] Every moment users are searching for information on the world wide web using search engines. However, users do not always get information they expect when searching the web. Searching the world wide for useful information web has become too difficult considering the dynamic and unstructured nature of the web.

Although there are very efficient general purpose search engines, but no such spatial search engine is available to search location based information efficiently and accurately.

Spatial Web Portal (SWP) utilizes the advantages of web portal techniques, such as easy to configure, share and integrate, is widely used by the Earth science community in Earth science data sharing and exchanging. These SWPs store a large amount of geospatial resources, including text files, raw and post-processed data, and various geospatial web services.

Popular SWPs include NASA's Earth Science Gateway (ESG), ESIP's Earth Information Exchange, FGDC's Geospatial One Stop Portal, NASA's Earth Observing System Clearinghouse, NASA's Global Change Master Directory and NOAA's National Climatic Data Center.

For Geospatial search engine, in order to fetch needed geospatial information on the web to the end user, the first important thing is to identify where the geospatial information is available. When supplied with a spatial query, a typical search engine only return web pages that include that place name involved.

GPSE, the response to a query generally is satisfying links of relevant html pages, which are easy to show for the end user. On the contrary, spatial information is location-referenced and is supposed to be rendered as a map.

However, the popular utilizations also stand out problems of how to find needed data from a variety of geospatial resources and how to visualize the data from multi-perspectives.

A large fraction of documents on W3 (World Wide Web) contains geo-spatial context[16], but conventional search engines treat the query place name in the same way as any other keyword and retrieve documents that include the specified name. Conventional approach may be sufficient for many users but there are many occasions when user is interested in documents that are related to the region of space as user specified in his query, but which might not actually include the place name. This could generally happen in the situation when there were documents that used alternative names, or referred places that were in or nearby the particular place. In the current work an approach for indexing the document along with its spatial information is being discussed that can help to search the spatial data as per ones need.

Spatial Search Engine aims to search the World Wide Web for geographic information to meet the users' needs automatically and conveniently.

Spatial search engines are specialized search engines primarily dedicated to retrieve geographical information through web technology[17]. They provide capabilities to query metadata records for related spatial data, and link directly to the online content of spatial data themselves.

II. Existing works

In past researchers have done some work in the area of geo-spatial search and developed few products. Google location search engines and Vicinity products etc are few examples of such products but not much information has been published[18]. A spatial search engine named spirit was developed which had the feature of searching for places in geographic context like near, in etc. The search engine only return places and is basically a keyword matching search engine where there is no scope of intelligently understanding the user's search query [1].

III. WHY ONLY LUCENE AND SOLR

- Apache Lucene is a free and open-source information retrieval software library, written by Doug Cutting in java and is supported by Apache Software Foundation and released under the Apache Software License.
- Apache Lucene is an open source Java-based search library providing Application Programming Interfaces for performing common search and search related tasks like indexing, querying, highlighting, language analysis and many others [25].
- One popular feature available in Solr is its ability to do location-based searching. This is most commonly implemented by indexing a field in each document containing a geographical point (a latitude and longitude), and then asking Solr at query time to filter out documents that do not fall within a specified radius of some other point.
- Much better and quicker term relevancy ranking, efficiently customizable at search time.
- Faster search performance for familiar terms or complicated and complex queries.
- More efficient indexing performance than Postgres.
- Solr has three inbuilt tools, namely geospatial boundary box, a geospatial filter and a geospatial distance function.

Solr supports location data for use in geospatial/spatial searches. Using spatial search, you can:

a. Index polygons ,points ,circles or other shapes
b. Filter search results by a circle or bounding box or by other shapes
c. Sort or boost scoring by relative area, or distance between points.
d. Generate a 2D grid of plane count numbers for point-plotting or heat map generation.

Solr is an open source search platform merged in the Apache Lucene project. Solr is written in Java and runs as a full text search server. It includes full-text search, hit highlighting, near real-time indexing, faceted search, database integration, dynamic clustering, rich document handling, and geospatial search. Other strengths are fault tolerant and scalability, replication, distributed indexing and load-balanced querying, and automated failover and recovery.

The spatial module is new to Lucene 4.The principle interface to the module is a Spatial Strategy which encapsulates an approach to indexing and searching based on shapes. Different Strategies have distinct features and performance profiles, which are documented at each Strategy implementation class level.

Solr is a well-known and widely used search engine that deals with geographical entities. Solr give us a search solution, well developed, fast and free, and the support of a large community of developers.

One can easily find Solr about the kind of a data a field contains by specifying its field type. Field type basically tells Solr how to explain the field and how to query it. When a document is added, Solr extracts the information in the document's field to add that information to an index and later this index can quickly be consulted to provide the matching documents as result for a given.
Solr represents features, helpful for spatial search. With the help of these features one can represent the spatial information along with textual information into index, can filter data by location while retrieving it from index, and can sort it based on distance. To incorporate all above mentioned requirement, SOLR has three inbuilt tools, namely a geospatial filter, geospatial boundary box and a geospatial distance function. With the help of geo-filter one can retrieve all relevant documents within distance from a given point, e.g. retrieving all documents within 10 km radius from a given latitude/longitude.

Spatial4j is a general purpose spatial / geospatial ASL licensed open-source Java library. Its core capabilities are: to provide common geospatially-aware shapes, to provide distance calculations and to read and write the shapes to strings.Spatial4j is largely centered on geometric shape implementations:Define geometric shapes, both classic Euclidean/Cartesian two-dimensional ones and geodetic (surface-of-sphere) ones. Some sample shapes are: point, rectangle, circle (aka point-distance), polygon, line string

- Compute the bounding box of a shape.
- Compute the intersection case between a pair of shapes, Intersection cases are intersects, contains, within, and disjoint.
- Compute the area of a shape.
- Compute the distance between points (plus other misc. distance / angle calculations).
- Calculate geohash string for a point and compute the rectangle for a geohash, (plus other misc. geohash calculations)

IV. MODEL FOR SEARCHING

For Geospatial search engine, in order to fetch needed geospatial information on the web to the end user, the first important thing is to identify where the geospatial information is [22].
Here is a model to take input as a raw data which in form of indexed by lucene and then connecting as a indexing pipeline with core solr (base on Lucene and Tika) . Front end GUI or web search site connected with API layers to Solr to retrieve spatial point, polygons and represent geo-data or relevant information geographical map. Spatial queries, such as range search and nearest neighbour

retrieval, involve only conditions on objects geometric properties [23].

Searcher: Given a query, it must quickly find nodes: single point (latitude, longitude), ways (collection of nodes), relations (logical grouping of ways).To find a large relevant dataset is normally done with inverted index and rank to return most relevant documents for efficient filtering probabilistic data structure(e.g. bloom filters).
Finding a large relevant subset is normally done with an inverted index of the corpus; ranking within that set to produce the most relevant documents, which then must be summarized for display

Indexer: Creates the inverted index from which the searcher extracts results. It uses Lucene storing indexes.

Database: Stores the document contents for indexing and later summarization by the searcher, along with information such as the link structure of the document space and the time each document was last fetched.

Fetcher: Requests web pages, parses them, and extracts links from them.

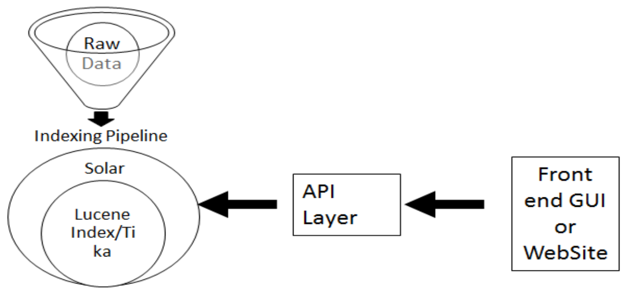

Figure 1: Model diagram

Indexing Geo-Spatial data

Lucene meets the scalability requirements for text indexing in Solr[8]. Lucene Spatial indexer used multiple approaches to index spatial data as class follows:

```
abstract class SpatialStrategy
 (5+ concrete implementations)
```

Figure 2: Listing for SerializedDVStrategy[14]

1 RecursivePrefixTreeStrategy(RPT):
It is most prominent,versatile grid based technique used in lucene.It uses Spatial4j library for shapes and distance calculation under which JTS topology suite library used for polygons.RPT uses Lucene's index as a PrefixTree.Thus it represents shapes as grid cells of varying precision by prefix. Example:-
Point→D, DR, DRT, DRT2, DRT2Y.
Polygon→ Many in list… 508 cells.

2 **SerializedDVStrategy:** It stores serialized geometry into Lucene BinaryDocValues. It is as accurate as the underlying geometry coordinates/shape. But it is a retrievable on a per document basis.

```
SpatialArgs args= new
SpatialArgs(INTERSECTS,point);
treeStrategy= new
RecursivePrefixTreeStrategy(grid,"geometry");
verifyStrategy= new
SerializedDVStrategy(ctx,"serialized_geometry");
Query treeQuery = new
ConstantScoreQuery(treeStrategy.makeFilter(args));
Query combinedQuery = new
FilteredQuery(treeQuery,verifyStrategy.makeFilter(args),
FilteredQuery.QUERY_FIRST_FILTER_STRATEGY);
```

Figure 2: Listing for SerializedDVStrategy[9]

3 **FlexPrefixTree:** A new SpatialPrefixTree which is more flexible than Goehash & Quad. It is configurable sub cells at each level: 4, 16, 64, 256. Internally uses an integer coordinate system where rectangle searches are particularly fast which has minimal floating conversion. Cells are always square or equal sides.

4 **BBoxSpatialStrategy:** Rectangles (Box's) only one value per filed. It support wide predicates such as equals, intersects, within, contains, disjoint. It is accurate such as 8-byte double floating pount and area overlap relevancy based on weight search result by a combination of query shape overlap and index shape overlap ratios [24].

Figure3: Solr BboxField [13]

5 **SolrBBoxField:** It search with overlap ratio ordering.

3.6 Searching

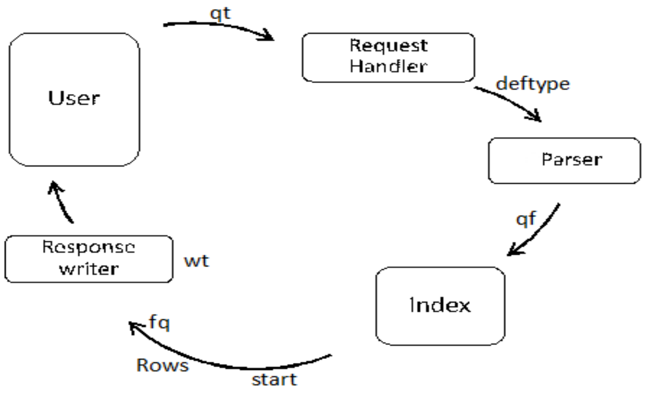

Figure 4: Core solr spatial search process[10]

qt: It selects a RequestHandler for a query using/select. DisMaxRequestHandler is used by default.
defType: It selects a query parser for the query. It used all configuration for RequestHandler.
qf: It selects which fields to query in the index.
start: It specifies an offset into the query results where the returned response should begin. It has default offset value is 0.
rows: It specifies the number of rows to be displayed at one time.
fq: It filters query by applying an additional query to the initial query's caches the results.
wt: It selects a response writer for formatting the query response.

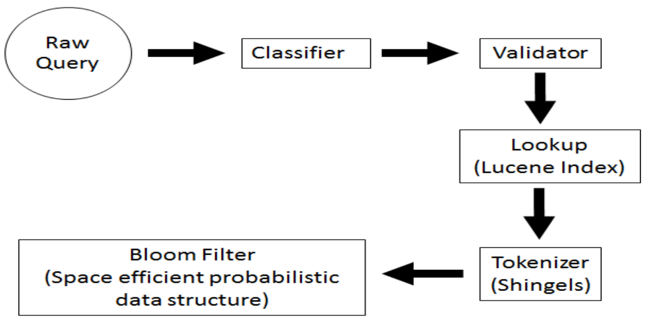

Figure 5: Searching with classification stages [11]

Raw query: It contains basic string query like "all malls near Rajiv chowk in Delhi".
Classifier: It identify which part of query string as possibility which part could mean geographical entity.
Validator: Validates all above classification.
Lookup: It acts like lucene index.
Tokenizer: It tokenizes queries into tokens or shingels (query=all, malls, near, Rajiv, chowk , in , Delhi).
Bloom Filters: It is a very efficient data structure for doing set membership. Indexed many item to bloom filters then check for whether all item inside bloom filters validate all classification. Probabilistic manner tell us the high degree of probability [26].

Searching near a single point

Solr's basic searching based upon a single point usually a latitude and longitude. This implementation supports a simple syntax for filtering based upon either a true circular radius or based upon a square (which is faster to calculate) with sides equal to the diameter of the true circular radius.

Defining the location fields

The first step in performing radius search is to create a field type in schema.xml to contain geographical location:

```
<fieldType> name="location"class="solr.LatLonType"
     subFieldSuffix="_coordinate" </fieldType>
```

Listing: Indexing location as latitude and longitude using subFieldSuffix type[28]

The LatLonType class, used for this location field definition, works for accepting a latitude/longitude pair in the form latitude, longitude and ultimately splitting the two coordinates and mapping them into separate fields. In order to map the latitude and longitude into two separate fields under the covers, those two fields will alose need to exist in the schema.xml. Add the dynamic field that ends with the subFieldSuffix specified in the fieldType definition :

```
<dynamicField name="_coordinate" type="tdouble"
     indexed="true" stored="false" />
```

Listing: Indexing location for geographical searching [12]

A dynamic field put in place for the location field to map the latitude and longitude coordinates into separately. Now, send all those documents into search engine that are left. Such in this case listing below:

```
<add>
   <doc>
        <field name="id">1</field>
        <field name="location">33.748,-84.391</field>
        <fieldname="city">Atlanta, GA</field>
        </doc>
        <doc>
        <field name="id">2</field>
        <field name="location">40.715,-74.007</field>
        <fieldname="city">New York, NY </field>
        </doc>
        <field name="id">3</field>
        <field name="location">37.775,-122.419</field>
        <fieldname="city">San Francisco, CA </field>
     </doc>
</add>
```

Listing: Indexing location for geographical searching [12]

Send the document to Solr by running the following command:

```
cd $solr 4.10/examples-doc/
java –Drupl=http://localhost:8983/solr/geospatial/update –jar
post.jar documents/geospatial.xml
```

Once documents are indexed in Solr, it's then possible to search upon location in various ways. It shows the search result for text vision of location with a query such as city: "New York, NY". This search outside of the borders of the city for which user are searching. This approach is problematic thus to enhance it to geospatial search as follows:-

```
http://localhost:8983/solr/geospatial/select?=*:*&
fq={!geofilt sfield=location pt="37.775,-122.419  d=20}
```

Geography and Bounding Box Filters

Geographical searching is to query for all documents within a specific distance of a given location. Solr contains a specific query parser named geofilt that takes a latitude/longitude coordinate, a location field and a maximum distance (in kilometers) and matches only documents falling within the geographical area specified.[27]The syntax for such a query searching for a 20 kilometers radius from San Francisco, CA, it would be as follows:

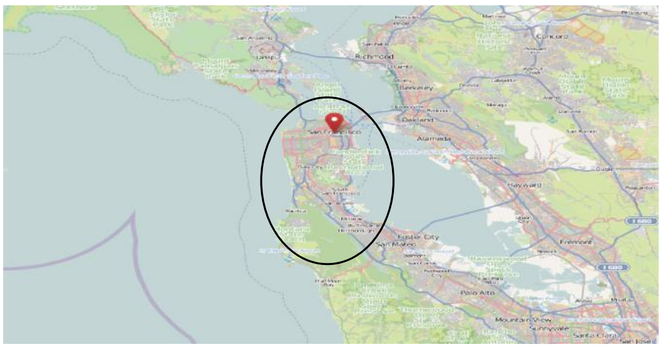

Figure 6: The Geographical area from which documents would be returned from a geofilt query requesting a distance of d=20km from the center of San Francisco, CA.[7]

To improve this location search Solr contains bbox(bounding box) query parser, making it easy to substitute the two within any geography filter:

```
http://localhost:8983/solr/geospatial/select?q=*:*&
fq={!bbox sfield=location pt=37.775,-122.410 d=20} [21]
```

Above performs a fast bounding box (bbox) filter to find geographically close location and how to apply a more accurate geofilt filter that will calculate distance for each document inside a bounding box and filter out which reside in radius range.
Solr sort the document by distance away or for calculated distance to be returned with remaining fields.

Advance Geospatial search

The Solr project grew out of the first author's experience developing lucene [20]. We have discussed simple single point geospatial search implementation. Solr upgraded to more advanced implementation that supports indexing much more single point per document. This advance feature added arbitrary polygons (as opposed to just points) and it supports indexing multiple points or shape per filed. If we search a point location then it index a single document with multiple latitude/longitude coordinates. At query time it return relevant document.

V. Proposal

This research discusses the spatial-aware search problem in current spatial web portals and based on that, it proposes a method to construct an intelligent search engine to improve the search efficiency. Our approach is to explore the lucene and Solr from spatial search point of view along with the support of geo-ontologies.
A geo-ontology play a key role in spatially-aware search engine, with regards to providing support for relevance ranking, query expansion, query disambiguation and web resource annotation. A geographical ontology is used to assist spatial search on the Internet

VI. Future Work

The faceted search feature can be added to the spatial search engine. Ranking can also be implemented in the search engine.
Knowledge base can be added to the search engine using geographical ontologies to enhance the accuracy of search results and better understanding of search query. Another feature that can be added is to re-search in the returned results. Just like when we ask the search engine to show all the cities near Lucknow having population greater than a million, later we perform another search in the results obtained from previous query to find cities among them who are near Delhi have high literacy rates.

VII. RESULT

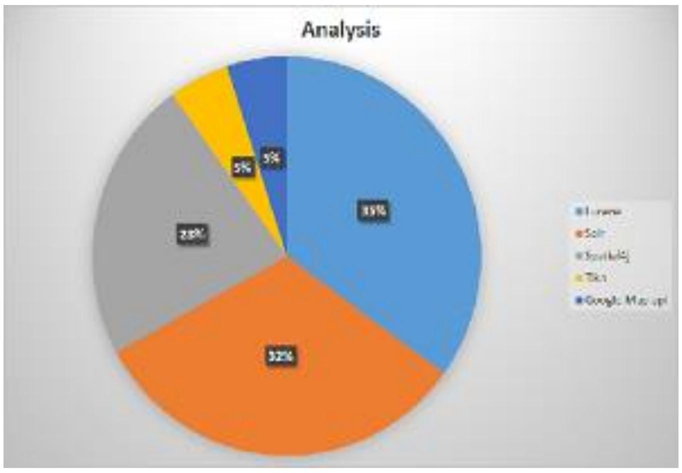

Figure 7: Component sizes in basic search setup

Above we are showing the basic resources used in search and similarly we will enhance it in our future work.
We implemented a basic search for bounding box circle using solr, lucene, geo-names api and google API. TIKA has been used to gather information. Lucene has been used to index the information specifically location (longitude ,latitude).

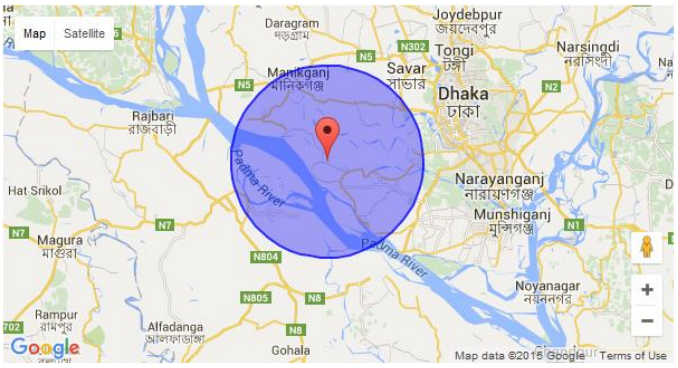

Figure 7**:** Map showing bounding box circle for our search

VIII. CONCLUSION

Solr is the key solution to an intelligent search engine when spatial data used with geographical ontology. They are efficient, more accurate and optimal solution for kind of spatial searching.

References

[1] Christopher B. Jones, Alia I. Abdelmoty, David Finch, Gaihua Fu, Subodh Vaid,"The SPIRIT Spatial Search Engine: Architecture, Ontologies and Spatial Indexing"

[2] Grant Ingersoll," Location-aware search with Apache Lucene and Solr"

[3] Gaihua Fu, Christopher B. Jones and Alia I. Abdelmoty," Building a Geographical Ontology for Intelligent Spatial Search on the Web"

[4] Divakar Yadav,Sonia Sanchez-Cuadrado,Jorge Morato, Juan Bautista Llorens Morillo,"An Approach for Spatial Search Using SOLR"

[5] Yuqi Bai , Chongjun Yang , Donglin Liu , Lingling Guo, "Spatial search engine – enabling the intelligent geographic information retrieval"

[6] Wenwen Li, Chaowei Yang ,"A Semantic Search Engine For Spatial Web Portals"

[7] Trey Grainger and Timothy Potter forward by Yonik Seely,"Solr in Action"

[8] Video conference of Lucene/Solr revolution by Lucidworks

[9] Presentation by th Climate Corporation presented of FOSS4G 2014

[10] http://www.edureka.co/apache-solr

[11] Lucene/Solr Revolution by Ishan Chattopadhyay, LucidWorks, www.lucenerevolution.org

[12] Trey Grainger and Timothy Potter forward by Yonik Seely,"Solr in Action" ISBN 9781617291029

[13] Beng Chin Ooi, Ron Sacks-Davis, Jiawei Han, "Indexing in spatial data"

[14] Video conference of Lucene/Solr revolution by Lucidworks 2013

[15] http://www.worldwidewebsize.com/

[16] Sergey Brin, Lawrence Page, The Anatomy of a Large-Scale Hypertextual Web Search Engine. Computer Networks and ISDN Systems, 30(1-7), pp.107-117

[17] Steve Lawrence, C. Lee Giles, Searching the Web: General and Scientific Information Access. IEEE Communications, 37(1), pp.116-122

[18] Steve Lawrence, C. Lee Giles, Searching the World Wide Web, Science, 280(5360), pp.98–100

[19] Location-aware search with Apache Lucene and Solr, Combine unstructured text and spatial data to enhance your search applications

[20] Rohit Khare, Doug Cutting, Kragen Sitaker, Adam Rifkin, Nutch: A Flexible and Scalable Open-Source Web Search Engine

[21] Trey Grainger and Timothy Potter forward by Yonik Seely,"Solr in Action" ISBN 9781617291029

[22] Yuqi BAI a, Chongjun YANG a, Donglin LIUa, Lingling GUOb –"SPATIAL SEARCH ENGINE – ENABLING THE INTELLIGENT GEOGRAPHIC INFORMATION RETRIEVAL"

[23] Sophiya.K, Sounderrajan.T - "Implements The Spatial Inverted Indexes To Perform Quick Search"

[24] Apache Solr Reference Guide 4.10-"Covering Apache Solr"

[25] Andrzej Białecki, Robert Muir, Grant Ingersoll - "Apache Lucene 4"

[26] Lucene/Solr Revolution by Ishan Chattopadhyay, LucidWorks, www.lucenerevolution.org

[27] Trey Grainger and Timothy Potter forward by Yonik Seely,"Solr in Action" ISBN 9781617291029

[28] Apache Solr Reference Guide 5-"Covering Apache Solr"